Package ‘ade4’

June 17, 2021

Version 1.7-17

Title Analysis of Ecological Data: Exploratory and Euclidean Methods in Environmental Sciences

Author Stéphane Dray <stephane.dray@univ-lyon1.fr>, Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>, and Jean Thioulouse <jean.thioulouse@univ-lyon1.fr>, with contributions from Thibaut Jombart, Sandrine Pavoine, Jean R. Lobry, Sébastien Ollier, Daniel Borcard, Pierre Legendre, Stéphanie Bougeard and Aurélie Siberchicot. Based on earlier work by Daniel Chessel.

Maintainer Aurélie Siberchicot <aurelie.siberchicot@univ-lyon1.fr>

Depends R (>= 2.10)

Imports graphics, grDevices, methods, stats, utils, MASS, pixmap, sp

Suggests ade4TkGUI, adegraphics, adephylo, ape, CircStats, deldir, lattice, spdep, splancs, waveslim, progress, foreach, parallel, doParallel, iterators

Description Tools for multivariate data analysis. Several methods are provided for the analysis (i.e., ordination) of one-table (e.g., principal component analysis, correspondence analysis), two-table (e.g., co-inertia analysis, redundancy analysis), three-table (e.g., RLQ analysis) and K-table (e.g., STATIS, multiple co-inertia analysis). The philosophy of the package is described in Dray and Dufour (2007) <doi:10.18637/jss.v022.i04>.

License GPL (>= 2)

URL http://pbil.univ-lyon1.fr/ADE-4/

BugReports https://github.com/sdray/ade4/issues

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-06-17 16:10:04 UTC

R topics documented:

  ade4-package .......................................................... 8
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>abouheif.eg</td>
<td>8</td>
</tr>
<tr>
<td>acacia</td>
<td>9</td>
</tr>
<tr>
<td>add.scatter</td>
<td>10</td>
</tr>
<tr>
<td>aminoacyl</td>
<td>13</td>
</tr>
<tr>
<td>amova</td>
<td>14</td>
</tr>
<tr>
<td>apis108</td>
<td>15</td>
</tr>
<tr>
<td>apqe</td>
<td>16</td>
</tr>
<tr>
<td>aravo</td>
<td>17</td>
</tr>
<tr>
<td>ardeche</td>
<td>18</td>
</tr>
<tr>
<td>area.plot</td>
<td>19</td>
</tr>
<tr>
<td>arrival</td>
<td>21</td>
</tr>
<tr>
<td>as.taxo</td>
<td>22</td>
</tr>
<tr>
<td>atlas</td>
<td>23</td>
</tr>
<tr>
<td>atya</td>
<td>25</td>
</tr>
<tr>
<td>avijons</td>
<td>26</td>
</tr>
<tr>
<td>avimedi</td>
<td>28</td>
</tr>
<tr>
<td>aviurba</td>
<td>29</td>
</tr>
<tr>
<td>bacteria</td>
<td>30</td>
</tr>
<tr>
<td>banque</td>
<td>31</td>
</tr>
<tr>
<td>baran95</td>
<td>34</td>
</tr>
<tr>
<td>bca</td>
<td>36</td>
</tr>
<tr>
<td>bca.coinertia</td>
<td>38</td>
</tr>
<tr>
<td>bca.rlq</td>
<td>39</td>
</tr>
<tr>
<td>between</td>
<td>40</td>
</tr>
<tr>
<td>bf88</td>
<td>42</td>
</tr>
<tr>
<td>bicenter.wt</td>
<td>43</td>
</tr>
<tr>
<td>bordeaux</td>
<td>44</td>
</tr>
<tr>
<td>bsetal97</td>
<td>44</td>
</tr>
<tr>
<td>buech</td>
<td>46</td>
</tr>
<tr>
<td>butterfly</td>
<td>47</td>
</tr>
<tr>
<td>bwca.dpcoa</td>
<td>48</td>
</tr>
<tr>
<td>cailliez</td>
<td>50</td>
</tr>
<tr>
<td>capitales</td>
<td>51</td>
</tr>
<tr>
<td>carni19</td>
<td>52</td>
</tr>
<tr>
<td>carni70</td>
<td>53</td>
</tr>
<tr>
<td>carniherbi49</td>
<td>54</td>
</tr>
<tr>
<td>casitas</td>
<td>55</td>
</tr>
<tr>
<td>chatcat</td>
<td>56</td>
</tr>
<tr>
<td>chats</td>
<td>57</td>
</tr>
<tr>
<td>chazeb</td>
<td>58</td>
</tr>
<tr>
<td>chevaine</td>
<td>59</td>
</tr>
<tr>
<td>chickenk</td>
<td>60</td>
</tr>
<tr>
<td>clementines</td>
<td>61</td>
</tr>
<tr>
<td>cnc2003</td>
<td>62</td>
</tr>
<tr>
<td>coinertia</td>
<td>63</td>
</tr>
<tr>
<td>coleo</td>
<td>65</td>
</tr>
<tr>
<td>combine.4thcorner</td>
<td>66</td>
</tr>
<tr>
<td>corkdist</td>
<td>68</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>corvus .......................................................... 69</td>
<td></td>
</tr>
<tr>
<td>costatis ........................................................... 70</td>
<td></td>
</tr>
<tr>
<td>costatis.randtest ................................................ 71</td>
<td></td>
</tr>
<tr>
<td>dagnelie.test ..................................................... 72</td>
<td></td>
</tr>
<tr>
<td>Deprecated functions .............................................. 74</td>
<td></td>
</tr>
<tr>
<td>deug ................................................................. 74</td>
<td></td>
</tr>
<tr>
<td>disc ................................................................. 75</td>
<td></td>
</tr>
<tr>
<td>discrimin ........................................................... 76</td>
<td></td>
</tr>
<tr>
<td>discrimin.coa ....................................................... 78</td>
<td></td>
</tr>
<tr>
<td>dist.binary .......................................................... 79</td>
<td></td>
</tr>
<tr>
<td>dist.dudi ............................................................ 80</td>
<td></td>
</tr>
<tr>
<td>dist.ktab ............................................................. 81</td>
<td></td>
</tr>
<tr>
<td>dist.neig ............................................................. 84</td>
<td></td>
</tr>
<tr>
<td>dist.prop ............................................................. 85</td>
<td></td>
</tr>
<tr>
<td>dist.quant ............................................................ 86</td>
<td></td>
</tr>
<tr>
<td>divc ................................................................. 88</td>
<td></td>
</tr>
<tr>
<td>divcmax .............................................................. 89</td>
<td></td>
</tr>
<tr>
<td>dotchart.phylog ..................................................... 90</td>
<td></td>
</tr>
<tr>
<td>dotcircle ............................................................ 92</td>
<td></td>
</tr>
<tr>
<td>doubs ................................................................. 93</td>
<td></td>
</tr>
<tr>
<td>dpcoa ................................................................. 94</td>
<td></td>
</tr>
<tr>
<td>dudi ................................................................. 96</td>
<td></td>
</tr>
<tr>
<td>dudi.acm ............................................................. 98</td>
<td></td>
</tr>
<tr>
<td>dudi.coa ............................................................. 100</td>
<td></td>
</tr>
<tr>
<td>dudi.dec ............................................................. 101</td>
<td></td>
</tr>
<tr>
<td>dudi.fca ............................................................. 102</td>
<td></td>
</tr>
<tr>
<td>dudi.hillsmith ...................................................... 104</td>
<td></td>
</tr>
<tr>
<td>dudi.mix ............................................................. 106</td>
<td></td>
</tr>
<tr>
<td>dudi.nsc ............................................................. 107</td>
<td></td>
</tr>
<tr>
<td>dudi.pca ............................................................. 108</td>
<td></td>
</tr>
<tr>
<td>dudi.pco ............................................................. 108</td>
<td></td>
</tr>
<tr>
<td>dunedata ............................................................. 110</td>
<td></td>
</tr>
<tr>
<td>ecg ................................................................. 112</td>
<td></td>
</tr>
<tr>
<td>ecomor ............................................................... 112</td>
<td></td>
</tr>
<tr>
<td>elec88 ............................................................... 113</td>
<td></td>
</tr>
<tr>
<td>escopage ............................................................. 115</td>
<td></td>
</tr>
<tr>
<td>euro123 ............................................................... 117</td>
<td></td>
</tr>
<tr>
<td>fission ............................................................... 118</td>
<td></td>
</tr>
<tr>
<td>foucart .............................................................. 119</td>
<td></td>
</tr>
<tr>
<td>fourthcorner ........................................................ 120</td>
<td></td>
</tr>
<tr>
<td>friday87 ............................................................. 122</td>
<td></td>
</tr>
<tr>
<td>fruits ................................................................. 125</td>
<td></td>
</tr>
<tr>
<td>gearymoran .......................................................... 126</td>
<td></td>
</tr>
<tr>
<td>ggtortoises .......................................................... 128</td>
<td></td>
</tr>
<tr>
<td>granulo ............................................................... 129</td>
<td></td>
</tr>
<tr>
<td>gridrowcol ........................................................... 131</td>
<td></td>
</tr>
<tr>
<td>hdpg ................................................................. 132</td>
<td></td>
</tr>
<tr>
<td>housetasks .......................................................... 133</td>
<td></td>
</tr>
<tr>
<td></td>
<td>134</td>
</tr>
</tbody>
</table>
R topics documented:

humDNAm ......................................................... 135
ichtyo ............................................................. 136
inertia.dudi ......................................................... 137
irishdata ........................................................... 138
is.euclid ............................................................ 140
julliot ............................................................... 141
jv73 ................................................................. 143
kcponds ............................................................. 144
kdist ................................................................. 146
kdist2ktab .......................................................... 148
kdisteuclid ......................................................... 149
kplot ................................................................. 150
kplot.foucart ...................................................... 151
kplot.mcoa ........................................................ 152
kplot.mfa .......................................................... 153
kplot.mfa .......................................................... 154
kplot.pta .......................................................... 155
kplot.sepan ....................................................... 157
krandtest .......................................................... 158
ktab ................................................................. 159
ktab.data.frame ................................................... 161
ktab.list.df ....................................................... 162
ktab.list.dudi ..................................................... 163
ktab.match2ktabs ................................................ 164
ktab.within ....................................................... 165
lascaux ............................................................. 166
lingoes ............................................................. 168
lizards ............................................................. 169
loocv.between ..................................................... 170
loocv.discrimin .................................................. 172
loocv.dudi ......................................................... 174
macaca ............................................................. 176
macon .............................................................. 177
macroloire ........................................................ 177
mafragh ........................................................... 179
mantel.randtest .................................................. 181
mantel.rtest ....................................................... 182
maples ............................................................. 183
mariages .......................................................... 184
mbpcaiv ............................................................ 185
mbpls .............................................................. 187
mcoa .............................................................. 189
mdpcoa ............................................................. 190
meau ............................................................... 193
meaudret .......................................................... 195
mfa ................................................................. 196
microsatt .......................................................... 198
mjrochet .......................................................... 199
R topics documented:

- mld .................................................. 200
- mollusc .............................................. 202
- monde84 ............................................ 203
- morphosport ....................................... 204
- mstree ............................................... 205
- multiblock .......................................... 206
- multispati .......................................... 207
- multispati.randtest .............................. 210
- multispati.rtest ................................... 212
- neig .................................................. 213
- newick,eg .......................................... 216
- newick2phylog ..................................... 217
- niche .................................................. 219
- nipals ............................................... 221
- njplot .............................................. 224
- olympic ............................................ 225
- oribatid ........................................... 226
- originality ....................................... 227
- orthobasis ........................................ 228
- ours ............................................... 231
- palm ............................................... 233
- pap .................................................. 234
- pcaiv .............................................. 235
- pcaivortho ........................................ 239
- pcoscaled ......................................... 241
- pcw .................................................. 242
- perthi02 .......................................... 243
- phylog ............................................. 244
- PI2newick ........................................ 246
- piosphere ........................................... 247
- plot.phylog ........................................ 248
- presid2002 ........................................ 250
- procella ........................................... 252
- procuste ........................................... 253
- procuste.randtest ............................... 256
- procuste.rtest .................................... 257
- pta .................................................. 258
- quasieuclid ....................................... 260
- randboot .......................................... 261
- randboot.multiblock ............................ 262
- randtest ........................................... 263
- randtest.amova ................................... 264
- randtest.between ................................. 265
- randtest.coinertia ............................... 266
- randtest.discrimin .............................. 267
- randtest.dpcoa ................................... 268
- randtest.pcaiv .................................... 270
- randxval .......................................... 271
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rankrock</td>
<td>272</td>
</tr>
<tr>
<td>reconst</td>
<td>272</td>
</tr>
<tr>
<td>rhizobium</td>
<td>274</td>
</tr>
<tr>
<td>rhone</td>
<td>275</td>
</tr>
<tr>
<td>rlq</td>
<td>276</td>
</tr>
<tr>
<td>rpjdl</td>
<td>279</td>
</tr>
<tr>
<td>rtest</td>
<td>280</td>
</tr>
<tr>
<td>rtest.between</td>
<td>281</td>
</tr>
<tr>
<td>rtest.discrimin</td>
<td>282</td>
</tr>
<tr>
<td>RV.rtest</td>
<td>283</td>
</tr>
<tr>
<td>RVdist.randtest</td>
<td>284</td>
</tr>
<tr>
<td>s.arrow</td>
<td>284</td>
</tr>
<tr>
<td>s.chull</td>
<td>286</td>
</tr>
<tr>
<td>s.class</td>
<td>287</td>
</tr>
<tr>
<td>s.corecircle</td>
<td>289</td>
</tr>
<tr>
<td>s.distri</td>
<td>291</td>
</tr>
<tr>
<td>s.hist</td>
<td>293</td>
</tr>
<tr>
<td>s.image</td>
<td>294</td>
</tr>
<tr>
<td>s.kde2d</td>
<td>296</td>
</tr>
<tr>
<td>s.label</td>
<td>297</td>
</tr>
<tr>
<td>s.logo</td>
<td>299</td>
</tr>
<tr>
<td>s.match</td>
<td>301</td>
</tr>
<tr>
<td>s.match.class</td>
<td>302</td>
</tr>
<tr>
<td>s.multinom</td>
<td>304</td>
</tr>
<tr>
<td>s.traject</td>
<td>306</td>
</tr>
<tr>
<td>s.value</td>
<td>308</td>
</tr>
<tr>
<td>santacatalina</td>
<td>310</td>
</tr>
<tr>
<td>sarcelles</td>
<td>311</td>
</tr>
<tr>
<td>scalewt</td>
<td>312</td>
</tr>
<tr>
<td>scatter</td>
<td>313</td>
</tr>
<tr>
<td>scatter.acm</td>
<td>314</td>
</tr>
<tr>
<td>scatter.coa</td>
<td>315</td>
</tr>
<tr>
<td>scatter.dudi</td>
<td>316</td>
</tr>
<tr>
<td>scatter.fca</td>
<td>317</td>
</tr>
<tr>
<td>scatterutil</td>
<td>318</td>
</tr>
<tr>
<td>sco.boxplot</td>
<td>320</td>
</tr>
<tr>
<td>sco.class</td>
<td>321</td>
</tr>
<tr>
<td>sco.distri</td>
<td>322</td>
</tr>
<tr>
<td>sco.gauss</td>
<td>324</td>
</tr>
<tr>
<td>sco.label</td>
<td>325</td>
</tr>
<tr>
<td>sco.match</td>
<td>326</td>
</tr>
<tr>
<td>sco.quant</td>
<td>328</td>
</tr>
<tr>
<td>score</td>
<td>329</td>
</tr>
<tr>
<td>score.acm</td>
<td>330</td>
</tr>
<tr>
<td>score.coa</td>
<td>331</td>
</tr>
<tr>
<td>score.mix</td>
<td>332</td>
</tr>
<tr>
<td>score.pca</td>
<td>333</td>
</tr>
<tr>
<td>seconde</td>
<td>334</td>
</tr>
<tr>
<td>Topics Documented</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
</tr>
<tr>
<td>sepan</td>
<td>335</td>
</tr>
<tr>
<td>skulls</td>
<td>336</td>
</tr>
<tr>
<td>statico</td>
<td>337</td>
</tr>
<tr>
<td>statico.krandtest</td>
<td>338</td>
</tr>
<tr>
<td>statis</td>
<td>339</td>
</tr>
<tr>
<td>steppe</td>
<td>341</td>
</tr>
<tr>
<td>supcol</td>
<td>342</td>
</tr>
<tr>
<td>supdist</td>
<td>343</td>
</tr>
<tr>
<td>suprow</td>
<td>345</td>
</tr>
<tr>
<td>suprow.pta</td>
<td>347</td>
</tr>
<tr>
<td>symbols.phylog</td>
<td>349</td>
</tr>
<tr>
<td>syndicats</td>
<td>350</td>
</tr>
<tr>
<td>t3012</td>
<td>351</td>
</tr>
<tr>
<td>table.cont</td>
<td>352</td>
</tr>
<tr>
<td>table.dist</td>
<td>353</td>
</tr>
<tr>
<td>table.paint</td>
<td>354</td>
</tr>
<tr>
<td>table.phylog</td>
<td>355</td>
</tr>
<tr>
<td>table.value</td>
<td>356</td>
</tr>
<tr>
<td>tarentaise</td>
<td>358</td>
</tr>
<tr>
<td>taxo.eg</td>
<td>359</td>
</tr>
<tr>
<td>testdim</td>
<td>360</td>
</tr>
<tr>
<td>testdim.multiblock</td>
<td>361</td>
</tr>
<tr>
<td>tintoodiel</td>
<td>362</td>
</tr>
<tr>
<td>tithonia</td>
<td>363</td>
</tr>
<tr>
<td>tortues</td>
<td>364</td>
</tr>
<tr>
<td>toxicity</td>
<td>365</td>
</tr>
<tr>
<td>triangle.class</td>
<td>366</td>
</tr>
<tr>
<td>triangle.plot</td>
<td>368</td>
</tr>
<tr>
<td>trichometeo</td>
<td>369</td>
</tr>
<tr>
<td>ungulates</td>
<td>370</td>
</tr>
<tr>
<td>uniquewt.df</td>
<td>371</td>
</tr>
<tr>
<td>variance.phylog</td>
<td>372</td>
</tr>
<tr>
<td>varipart</td>
<td>374</td>
</tr>
<tr>
<td>vegif</td>
<td>376</td>
</tr>
<tr>
<td>veuvage</td>
<td>377</td>
</tr>
<tr>
<td>wca</td>
<td>378</td>
</tr>
<tr>
<td>wca.coinertia</td>
<td>379</td>
</tr>
<tr>
<td>wca.rlq</td>
<td>381</td>
</tr>
<tr>
<td>westafrica</td>
<td>382</td>
</tr>
<tr>
<td>within</td>
<td>384</td>
</tr>
<tr>
<td>withinpca</td>
<td>386</td>
</tr>
<tr>
<td>witwit.coa</td>
<td>387</td>
</tr>
<tr>
<td>woangers</td>
<td>388</td>
</tr>
<tr>
<td>worksurv</td>
<td>390</td>
</tr>
<tr>
<td>yanomama</td>
<td>392</td>
</tr>
<tr>
<td>zealand</td>
<td>393</td>
</tr>
</tbody>
</table>

**Index** 395
The ade4 package

Description

This package is developed in the Biometry and Evolutionary Biology Lab (UMR CNRS 5558) - University Lyon 1. It contains Data Analysis functions to analyse Ecological and Environmental data in the framework of Euclidean Exploratory methods, hence the name ade4.

ade4 is characterized by (1) the implementation of graphical and statistical functions, (2) the availability of numerical data, (3) the redaction of technical and thematic documentation and (4) the inclusion of bibliographic references.

To cite ade4, please use citation("ade4").

Author(s)

Stéphane Dray, Anne-Béatrice Dufour, and Jean Thioulouse. Contributions from Daniel Borcard, Stéphanie Bougeard, Thibaut Jombart, Pierre Legendre, Jean R. Lobry, Sébastien Ollier, Sandrine Pavoine and Auréliе Siberchicot. Based on earlier work by Daniel Chessel.

References


See ade4 website: http://pbil.univ-lyon1.fr/ADE-4/

See Also

ade4TkGUI, adegenet, adehabitat, adegraphics

Phylogenies and quantitative traits from Abouheif

Description

This data set gathers three phylogenies with three sets of traits as reported by Abouheif (1999).

Usage

data(abouheif.eg)
**Format**

`abouheif.eg` is a list containing the 6 following objects:

- `tre1` is a character string giving the first phylogenetic tree made up of 8 leaves.
- `vec1` is a numeric vector with 8 values.
- `tre2` is a character string giving the second phylogenetic tree made up of 7 leaves.
- `vec2` is a numeric vector with 7 values.
- `tre3` is a character string giving the third phylogenetic tree made up of 15 leaves.
- `vec3` is a numeric vector with 15 values.

**Source**

Data taken from the phylogenetic independence program developed by Ehab Abouheif.

**References**


**Examples**

```r
data(abouheif.eg)
par(mfrow=c(2,2))
symbols.phylog(newick2phylog(abouheif.eg$tre1), abouheif.eg$vec1, sub = "Body Mass (kg)", csi = 2, csub = 2)
symbols.phylog(newick2phylog(abouheif.eg$tre2), abouheif.eg$vec2, sub = "Body Mass (kg)", csi = 2, csub = 2)
dotchart.phylog(newick2phylog(abouheif.eg$tre1), abouheif.eg$vec1, sub = "Body Mass (kg)", cdot = 2, cnod = 1, possub = "topleft", csub = 2, ceti = 1.5)
dotchart.phylog(newick2phylog(abouheif.eg$tre2), abouheif.eg$vec2, sub = "Body Mass (kg)", cdot = 2, cnod = 1, possub = "topleft", csub = 2, ceti = 1.5)
par(mfrow = c(1,1))
w.phy=newick2phylog(abouheif.eg$tre3)
dotchart.phylog(w.phy,abouheif.eg$vec3, clabel.n = 1)
```

---

**acacia**

Spatial pattern analysis in plant communities

**Description**

Counts of individuals of *Acacia ehrenbergiana* from five parallel transects of 32 quadrats.

**Usage**

data(acacia)
add.scatter

Add graphics to an existing plot

Description

add.scatter is a function which defines a new plot area within an existing plot and displays an additional graphic inside this area. The additional graphic is determined by a function which is the first argument taken by add.scatter. It can be used in various ways, for instance to add a screeplot to an ordination scatterplot (add.scatter.eig).

The function add.scatter.eig uses the following colors: black (represented axes), grey (axes retained in the analysis) and white (others).
Usage

add.scatter(func, posi = c("bottomleft", "bottomright", "topleft", "topright"),
    ratio = 0.2, inset = 0.01, bg.col = 'white')
add.scatter.eig(w, nf = NULL, xax, yax, posi = "bottomleft", ratio =
    .25, inset = 0.01, sub = "Eigenvalues", csub = 2 * ratio)

Arguments

func       an - evaluated - function producing a graphic
posi       a character vector (only its first element being considered) giving the position of
            the added graph. Possible values are "bottomleft" (="bottom"), "bottomright", "topleft"
            (="top"), "topright", and "none" (no plot).
ratio      the size of the added graph in proportion of the current plot region
inset      the inset from which the graph is drawn, in proportion of the whole plot region.
            Can be a vector of length 2, giving the inset in x and y. If atomic, same inset is
            used in x and y
bg.col     the color of the background of the added graph
w          numeric vector of eigenvalues
nf         the number of retained factors, NULL if not provided
xax        first represented axis
yax        second represented axis
sub        title of the screeplot
csub       size of the screeplot title

Details

add.scatter uses par("plt") to redefine the new plot region. As stated in par documenta-

tion, this produces to (sometimes surprising) interactions with other parameters such as "mar".
In particular, such interactions are likely to reset the plot region by default which would cause
the additional graphic to take the whole plot region. To avoid such inconvenient, add par([other
options],plt=par("plt")) when using par in your graphical function (argument func).

Value

The matched call (invisible).

Author(s)

Thibaut.Jombart <t.jombart@imperial.ac.uk>

See Also

scatter
Examples

```r
data(microsatt)
w <- dudi.coa(data.frame(t(microsatt$tab)), scann = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  a1 <- rnorm(100)
b1 <- s1d.barchart(sort(a1), p1d.horizontal = FALSE, plot = FALSE)
h1 <- s1d.hist(a1, pgrid.draw = FALSE, porigin.draw = FALSE, pbackground.col = "grey",
               plot = FALSE, ppoly.col = "white", ppoly.alpha = 1)
g1 <- insert(h1, b1, posi = "topleft", plot = FALSE)

  a2 <- rnorm(100)
b2 <- s1d.barchart(sort(a2), p1d.horizontal = FALSE, plot = FALSE)
h2 <- s1d.hist(a2, pgrid.draw = FALSE, porigin.draw = FALSE, pbackground.col = "grey",
               plot = FALSE, ppoly.col = "white", ppoly.alpha = 1)
g2 <- insert(h2, b2, posi = "topleft", inset = c(0.25, 0.01), plot = FALSE)

  a3 <- rnorm(100)
b3 <- s1d.barchart(sort(a3), p1d.horizontal = FALSE, plot = FALSE)
h3 <- s1d.hist(a3, pgrid.draw = FALSE, porigin.draw = FALSE, pbackground.col = "grey",
               plot = FALSE, ppoly.col = "white", ppoly.alpha = 1)
g3 <- insert(h3, b3, posi = "bottomleft", inset = 0.4, ratio = 0.2, plot = FALSE)

  a4 <- rnorm(100)
b4 <- s1d.barchart(sort(a4), p1d.horizontal = FALSE, plot = FALSE)
h4 <- s1d.hist(a4, pgrid.draw = FALSE, porigin.draw = FALSE, pbackground.col = "grey",
               plot = FALSE, ppoly.col = "white", ppoly.alpha = 1)
g4 <- insert(h3, b3, posi = "bottomright", ratio = 0.3, plot = FALSE)

  G1 <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2), plot = TRUE)

  g5 <- s.label(w$co, plot = FALSE)
g6 <- plotEig(w$eig, w$n, psub = list(text = "Eigenvalues"),
               pbackground = list(box = TRUE), plot = FALSE)
  G2 <- insert(g6, g5, posi = "bottomright", ratio = 0.25)
} else {
  par(mfrow=c(2,2))
f1 <- function(a){
  opar=par("mar","xaxt","yaxt","plt")
on.exit(par(opar))
  par(mar=rep(.1,4),xaxt="n",yaxt="n",plt=par("plt"))
  hist(a,xlab="",ylab="",main="",col="white",proba=TRUE)
  lines(seq(-4,4,le=50),dnorm(seq(-4,4,le=50)),col="red")
}

  a <- rnorm(100)
  barplot(sort(a))
  add.scatter(f1(a),posi="topleft",bg.col="grey")

  a <- rnorm(100)
```

barplot(sort(a))
add.scatter(f1(a),posi="topleft",bg.col="grey",inset=c(.25,.01))

a <- rnorm(100)
barplot(sort(a))
add.scatter(f1(a),posi="topleft",bg.col="grey",inset=.25,ratio=.1)

a <- rnorm(100)
barplot(sort(a))
add.scatter(f1(a),posi="bottomright",bg.col="grey",ratio=.3)
par(mfrow=c(1,1))
s.label(w$co)
add.scatter.eig(w$eig,w$nf,posi="bottomright",1,2)
}

<table>
<thead>
<tr>
<th>aminoacyl</th>
<th>Codon usage</th>
</tr>
</thead>
</table>

**Description**

aminoacyl is a list containing the codon counts of 36 genes encoding yeast aminoacyl-tRNA-synthetase(S.Cerevisiae).

**Usage**

data(aminoacyl)

**Format**

aminoacyl is a list containing the 5 following objects:

- **genes**: a vector giving the gene names.
- **localisation**: a vector giving the cellular localisation of the proteins (M = mitochondrial, C = cytoplasmic, I = indetermined, CI = cyto and mito).
- **codon**: a vector containing the 64 triplets.
- **AA**: a factor giving the amino acid names for each codon.
- **usage.codon**: a dataframe containing the codon counts for each gene.

**Source**

Data prepared by D. Charif <Delphine.Charif@versailles.inra.fr> starting from:
http://www.expasy.org/sprot/

**References**

Examples

data(amoacyl)
amoacyl$genes
amoacyl$usage.codon
dudi.coa(amoacyl$usage.codon, scannf = FALSE)

Description

The analysis of molecular variance tests the differences among population and/or groups of populations in a way similar to ANOVA. It includes evolutionary distances among alleles.

Usage

amova(samples, distances, structures)

## S3 method for class 'amova'
print(x, full = FALSE, ...)

Arguments

samples a data frame with haplotypes (or genotypes) as rows, populations as columns and abundance as entries

distances an object of class dist computed from Euclidean distance. If distances is null, equidistances are used.

structures a data frame containing, in the jth row and the kth column, the name of the group of level k to which the jth population belongs

x an object of class amova

full a logical value indicating whether the original data (‘distances’, ‘samples’, ‘structures’) should be printed

... further arguments passed to or from other methods

Value

Returns a list of class amova

call call

results a data frame with the degrees of freedom, the sums of squares, and the mean squares. Rows represent levels of variability.

componentsofcovariance a data frame containing the components of covariance and their contribution to the total covariance

statphi a data frame containing the phi-statistics
apis108

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>

References


See Also

randtest.amova

Examples

data(humDNAm)
amovahum <- amova(humDNAm$samples, sqrt(humDNAm$distances), humDNAm$structures)
amovahum

apis108

Allelic frequencies in ten honeybees populations at eight microsatellites loci

Description

This data set gives the occurrences for the allelic form on 8 loci in 10 populations of honeybees.

Usage

data(apis108)

Format

A data frame containing 180 rows (allelic forms on 8 loci) and 10 columns (populations of honeybees: El.Hermel, Al.Hoceima, Nimba, Celinda, Pretoria, Chalkidiki, Forli, Valenciennes, Umea and Seville).

Source


Examples

data(apis108)
str(apis108)
names(apis108)
Apportionment of Quadratic Entropy

Description

The hierarchical apportionment of quadratic entropy defined by Rao (1982).

Usage

\[
\text{apqe}(\text{samples, dis = NULL, structures})
\]

## S3 method for class 'apqe'

\[
\text{print}(x, \text{ full = FALSE, ...})
\]

Arguments

- **samples**: a data frame with haplotypes (or genotypes) as rows, populations as columns and abundance or presence-absence as entries
- **dis**: an object of class dist computed from Euclidean distance. If dis is null, equidistances are used.
- **structures**: a data frame that contains, in the jth row and the kth column, the name of the group of level k to which the jth population belongs
- **x**: an object of class apqe
- **full**: a logical value that indicates whether the original data ('distances', 'samples', 'structures') should be printed
- **...**: further arguments passed to or from other methods

Value

Returns a list of class apqe

- **call**: call
- **results**: a data frame that contains the components of diversity.

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>

References


aravo

Examples

data(ecomor)
ecomor.phylog <- taxo2phylog(ecomor$taxo)
apqe(ecomor$habitat, ecomor.phylog$Wdist)

aravo Distribution of Alpine plants in Aravo (Valloire, France)

Description

This dataset describe the distribution of 82 species of Alpine plants in 75 sites. Species traits and environmental variables are also measured.

Usage

data(aravo)

Format

aravo is a list containing the following objects:

spe is a data.frame with the abundance values of 82 species (columns) in 75 sites (rows).
env is a data.frame with the measurements of 6 environmental variables for the sites.
traits is data.frame with the measurements of 8 traits for the species.
spe.names is a vector with full species names.

Details

The environmental variables are:

Aspect Relative south aspect (opposite of the sine of aspect with flat coded 0)
Slope Slope inclination (degrees)
Form Microtopographic landform index: 1 (convexity); 2 (convex slope); 3 (right slope); 4 (concave slope); 5 (concavity)
Snow Mean snowmelt date (Julian day) averaged over 1997-1999
PhysD Physical disturbance, i.e., percentage of unvegetated soil due to physical processes
ZoogD Zoogenic disturbance, i.e., quantity of unvegetated soil due to marmot activity: no; some; high

The species traits for the plants are:

Height Vegetative height (cm)
Spread Maximum lateral spread of clonal plants (cm)
Angle Leaf elevation angle estimated at the middle of the lamina
Area Area of a single leaf
Thick Maximum thickness of a leaf cross section (avoiding the midrib)
SLA Specific leaf area
Nmass Mass-based leaf nitrogen content
Seed Seed mass
Source


Examples

data(aravo)
coa1 <- dudi.coa(aravo$spe, scannf = FALSE, nf = 2)
dudienv <- dudi.hillsmith(aravo$env, scannf = FALSE, nf = 2, row.w = coa1$lw)
duditrait <- dudi.pca(aravo$traits, scannf = FALSE, nf = 2, row.w = coa1$cw)
rlq1 <- rlq(dudienv, coa1, duditrait, scannf = FALSE, nf = 2)
plot(rlq1)

---

**ardeche**  

*Fauna Table with double (row and column) partitioning*

Description

This data set gives information about species of benthic macroinvertebrates in different sites and dates.

Usage

data(ardeche)

Format

ardeche is a list with 6 components.

- **tab** is a data frame containing fauna table with 43 species (rows) and 35 samples (columns).
- **row.blocks** is a vector containing the repartition of species in the 4 groups defining the species order.
- **dat.fac** is a date factor for samples (6 dates).
- **sta.fac** is a site factor for samples (6 sites).
- **esp.fac** is a species order factor (Ephemeroptera, Plecoptera, Coleoptera, Trichoptera).

Details

The columns of the data frame *ardeche* $tab$ define the samples by a number between 1 and 6 (the date) and a letter between A and F (the site).

Source

Examples

data(ardeche)
dudi1 <- dudi.coa(ardeche$tab, scan = FALSE)
s.class(dudi1$co, ardeche$dat.fac)
if(adegraphicsLoaded()) {
  s.label(dudi1$co, plab.cex = 0.5, add = TRUE)
} else {
  s.label(dudi1$co, clab = 0.5, add.p = TRUE)
}

area.plot

Graphical Display of Areas

Description

'area' is a data frame with three variables.
The first variable is a factor defining the polygons.
The second and third variables are the xy coordinates of the polygon vertices in the order where they are found.

area.plot: grey levels areas mapping
poly2area takes an object of class 'polylist' (maptools package) and returns a data frame of type area.
area2poly takes an object of type 'area' and returns a list of class 'polylist'
area2link takes an object of type 'area' and returns a proximity matrix which terms are given by the length of the frontier between two polygons.
area.util.contour, area.util.xy and area.util.class are three utility functions.

Usage

area.plot(x, center = NULL, values = NULL, graph = NULL, lwdgraph = 2,
  nclasslegend = 8, clegend = 0.75, sub = "", csub = 1,
  possub = "topleft", cpoint = 0, label = NULL, clabel = 0, ...)

area2poly(area)
poly2area(polys)
area2link(area)
area.util.contour(area)
area.util.xy(area)

Arguments

x a data frame with three variables
center a matrix with the same row number as x and two columns, the coordinates of polygone centers. If NULL, it is computed with area.util.xy
values if not NULL, a vector which values will be mapped to grey levels. The values must be in the same order as the values in unique(x[,1])
graph     if not NULL, graph is a neighbouring graph (object of class "neig") between polygons
lwdgraph  a line width to draw the neighbouring graph
nclasslegend if value not NULL, a number of classes for the legend
clegend    if not NULL, a character size for the legend, used with \texttt{par("cex")*clegend}
sub        a string of characters to be inserted as sub-title
csub       a character size for the sub-titles, used with \texttt{par("cex")*csub}
possub     a string of characters indicating the sub-titles position ("topleft", "topright", "bottomleft", "bottomright")
cpoint     if positive, a character size for drawing the polygons vertices (check up), used with \texttt{par("cex")*cpoint}
label      if not NULL, by default the levels of the factor that define the polygons are used as labels. To change this value, use label. These labels must be in the same order than \texttt{unique(x.area[,1])}
clabel     if not NULL, a character size for the polygon labels, used with \texttt{par("cex")*clabel}
polys      a list belonging to the 'polylist' class in the spdep package
area       a data frame of class 'area'
...        further arguments passed to or from other methods

Value

\texttt{poly2area} returns a data frame 'factor,x,y'.
\texttt{area2poly} returns a list of class polylist.

Author(s)

Daniel Chessel

Examples

data(elec88)
par(mfrow = c(2, 2))
area.plot(elec88$area, cpoint = 1)
area.plot(elec88$area, lab = elec88$lab$dep, clab = 0.75)
area.plot(elec88$area, clab = 0.75)
# elec88$neig <- neig(area = elec88$area)
area.plot(elec88$area, graph = elec88$neig, sub = "Neighbourhood graph", possub = "topright")
par(mfrow = c(1, 1))

## Not run:
par(mfrow = c(3, 3))
for(i in 1:9) {
  x <- elec88$tab[,i]
  area.plot(elec88$area, val = x, sub = names(elec88$tab)[i], csub = 3, cleg = 1.5)
}
par(mfrow = c(1, 1))

if(!adegraphicsLoaded()) {
  data(irishdata)
  w <- irishdata$area[c(42:53, 18:25), ]
  w$poly <- as.factor(as.character(w$poly))
  area.plot(w, clab = 2)
  points(68, 59, pch = 20, col = "red", cex = 3)
  points(68, 35, pch = 20, col = "red", cex = 3)
  points(45, 12, pch = 20, col = "red", cex = 3)
  sqrt((59 - 35) ^ 2) + sqrt((68 - 45) ^ 2 + (35 - 12) ^ 2)
  area2link(w)
}

## End(Not run)

data(irishdata)
par(mfrow = c(2, 2))
w <- ade4:::area.util.contour(irishdata$area)
xy <- ade4:::area.util.xy(irishdata$area)
area.plot(irishdata$area, cpoint = 1)
apply(w, 1, function(x) segments(x[1], x[2], x[3], x[4], lwd = 3))
area.plot(irishdata$area, clabel = 1)
s.label(xy, area = irishdata$Area, incl = FALSE, clab = 0, cpoi = 3, addax = FALSE, contour = w)
s.label(xy, area = irishdata$Area, incl = FALSE, addax = FALSE, contour = w)
par(mfrow = c(1, 1))

if(!adegraphicsLoaded()) {
  data(irishdata)
  par(mfrow = c(2, 2))
w <- ade4:::area.util.contour(irishdata$area)
xy <- ade4:::area.util.xy(irishdata$area)
area.plot(irishdata$area, cpoint = 1)
apply(w, 1, function(x) segments(x[1], x[2], x[3], x[4], lwd = 3))
area.plot(irishdata$area, clabel = 1)
s.label(xy, area = irishdata$Area, incl = FALSE, clab = 0, cpoi = 3, addax = FALSE, contour = w)
s.label(xy, area = irishdata$Area, incl = FALSE, addax = FALSE, contour = w)
par(mfrow = c(1, 1))
}

## End(Not run)

data(irishdata)
w <- irishdata$area[c(42:53, 18:25), ]
w$poly <- as.factor(as.character(w$poly))
area.plot(w, clab = 2)
points(68, 59, pch = 20, col = "red", cex = 3)
points(68, 35, pch = 20, col = "red", cex = 3)
points(45, 12, pch = 20, col = "red", cex = 3)
sqrt((59 - 35) ^ 2) + sqrt((68 - 45) ^ 2 + (35 - 12) ^ 2)
area2link(w)

arrivals

Arrivals at an intensive care unit
Description

This data set gives arrival times of 254 patients at an intensive care unit during one day.

Usage

data(arrival)

Format

arrival is a list containing the 2 following objects:

times is a vector giving the arrival times in the form HH:MM
hours is a vector giving the number of arrivals per hour for the day considered

Source

Data taken from the Oriana software developed by Warren L. Kovach <sales@kovcomp.com> starting from https://www.kovcomp.co.uk/oriana/index.html.

References


Examples

data(arrival)
dotcircle(arrival$hours, pi/2 + pi/12)

---

**as.taxo**

**Taxonomy**

Description

The function `as.taxo` creates an object of class `taxo` that is a sub-class of `data.frame`. Each column of the data frame must be a factor corresponding to a level $j$ of the taxonomy (genus, family, ...). The levels of factor $j$ define some classes that must be completely included in classes of factor $j+1$.

A factor with exactly one level is not allowed. A factor with exactly one individual in each level is not allowed. The function `dist.taxo` compute taxonomic distances.

Usage

as.taxo(df)
dist.taxo(taxo)

Arguments

df a data frame
taxo a data frame of class taxo
Value

as.taxo returns a data frame of class taxo. dist.taxo returns a numeric of class dist.

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

See Also

taxo2phylog to transform an object of class taxo into an object of class phylog

Examples

data(taxo.eg)
tax <- as.taxo(taxo.eg[[1]])
tax.phy <- taxa2phylog(as.taxo(taxo.eg[[1]]),add.tools=TRUE)
par(mfrow = c(1,2))
plot(tax.phy, clabel.l = 1.25, clabel.n = 1.25, f = 0.75)
plot(taxo2phylog(as.taxo(taxo.eg[[1]][sample(15),]),
clabel.l = 1.25, clabel.n = 1.25, f = 0.75)
par(mfrow = c(1,1))
all(dist.taxo(tax)==tax.phy$Wdist)

---

atlas

Small Ecological Dataset

Description

atlas is a list containing three kinds of information about 23 regions (The French Alps): geographical coordinates, meteorology and bird presences.

Usage

data(atlas)

Format

atlas is a list of 9 components:

area is a convex hull of 23 geographical regions.
xy are the coordinates of the region centers and altitude (in meters).
names.district is a vector of region names.
meteo is a data frame with 7 variables: min and max temperature in january; min and max temperature in july; january, july and total rainfalls.
birds is a data frame with 15 variables (species).
contour is a data frame with 4 variables (x1, y1, x2, y2) for the contour display of The French Alps.

alti is a data frame with 3 variables altitude in percentage [0,800], [800,1500] and [1500,5000].

Spatial is the map of the 23 regions of The French Alps (an object of the class SpatialPolygons of sp).

Spatial.contour is the contour of the map of the 23 regions of the French Alps (an object of the class SpatialPolygons of sp).

Source

Extract from:


Examples

data(atlas)
if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g11 <- s.Spatial(atlas$Spatial, pSp.col = "white", plot = FALSE)
    g12 <- s.label(atlas$area[, 2:3], plabels.cex = 0, plot = FALSE)
    g1 <- superpose(g11, g12, plot = FALSE)
    g2 <- s.label(atlas$xy, lab = atlas$names.district, Sp = atlas$Spatial,
                  pgrid.dra = FALSE, pSp.col = "white", plot = FALSE)
    obj3 <- sp::SpatialPolygonsDataFrame(Sr = atlas$Spatial, data = atlas$meteo)
    g3 <- s.Spatial(obj3[, 1], nclass = 12, psub = list(position = "topleft",
                          text = "Temp Mini January", cex = 2), plot = FALSE)
    g4 <- s.corcircle((dudi.pca(atlas$meteo, scann = FALSE)$co), plabels.cex = 1, plot = FALSE)
    G1 <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
    obj5 <- sp::SpatialPolygonsDataFrame(Sr = atlas$Spatial,
                                         data = dudi.pca(atlas$meteo, scann = FALSE)$li)
    g5 <- s.Spatial(obj5[, 1], nclass = 12, psub = list(position = "topleft",
                                         text = "Principal Component Analysis analysis", cex = 1.5), plot = FALSE)
    coal <- dudi.coa(atlas$birds, scann = FALSE, nf = 1)
    obj6 <- sp::SpatialPolygonsDataFrame(Sr = atlas$Spatial, data = coal$li)
    g6 <- s.Spatial(obj6[, 1], nclass = 12, psub = list(position = "topleft",
                                         text = "Correspondence analysis", cex = 1.5), plot = FALSE)
    g7 <- s.value(atlas$xy, coal$Axis1, Sp = atlas$Spatial.contour, ppoints.cex = 2,
                  porigin.include = FALSE, paxes.draw = FALSE, pSp.col = "white", plot = FALSE)
    g8 <- triangle.label(atlas$alti, plabels.cex = 0, plot = FALSE)
    G2 <- ADEgS(list(g5, g6, g7, g8), layout = c(2, 2))
  }
} else {
  op <- par(no.readonly = TRUE)
  par(mfrow = c(2, 2))
  area.plot(atlas$area, cpoin = 1.5)
  area.plot(atlas$area, lab = atlas$names.district, clab = 1)
  x <- atlas$meteo$mini.jan
names(x) <- row.names(atlas$meteo)
area.plot(atlas$area, val = x, ncl = 12, sub = "Temp Mini January", csub = 2, cleg = 1)
s.corcircle((dudi.pca(atlas$meteo, scann = FALSE)$co), clab = 1)

area.plot(atlas$area, val = dudi.pca(atlas$meteo, scann = FALSE)$li[, 1], ncl = 12,
    sub = "Principal Component Analysis analysis", csub = 1.5, cleg = 1)
birds.coa <- dudi.coa(atlas$birds, sca = FALSE, nf = 1)
x <- birds.coa$li$Axis1
area.plot(atlas$area, val = x, ncl = 12, sub = "Correspondence analysis", csub = 1.5, cleg = 1)

s.value(atlas$xy, x, contour = atlas$contour, csi = 2, incl = FALSE, addax = FALSE)
triangle.plot(atlas$alti)
par(op)
par(mfrow = c(1, 1))

---

**aty a**

*Genetic variability of Cacadors*

**Description**

This data set contains information about genetic variability of *Atya innocous* and *Atya scabra* in Guadeloupe (France).

**Usage**

data(atya)

**Format**

aty contains the following components:

- **xy** a data frame with the coordinates of the 31 sites
- **gen** a data frame with 22 variables collected on 31 sites
- **neig** an object of class neig
- **nb** a neighborhood object (class nb defined in package spdep)

**Source**

Examples

```r
## Not run:
data(atya)
if(requireNamespace("pixmap", quietly = TRUE)) {
  atya.digi <- pixmap::read.pnm(system.file("pictures/atyadigi.pnm",
    package = "ade4")
  atya.carto <- pixmap::read.pnm(system.file("pictures/atyacarto.pnm",
    package = "ade4")
  par(mfrow = c(1, 2))
pixmap:::plot(atya.digi)
pixmap:::plot(atya.carto)
points(atya$xy, pch = 20, cex = 2)
}
if(requireNamespace("spdep", quietly = TRUE)) {
  plot(neig2nb(atya$neig), atya$xy, col = "red", add = TRUE, lwd = 2)
  par(mfrow = c(1,1))
}
## End(Not run)
```

---

**avijons**

*Bird species distribution*

**Description**

This data set contains information about spatial distribution of bird species in a zone surrounding the river Rhône near Lyon (France).

**Usage**

```r
data(avijons)
```

**Format**

- **avijons** is a list with the following components:
  - **xy** a data frame with the coordinates of the sites
  - **area** an object of class area
  - **fau** a data frame with the abundance of 64 bird species in 91 sites
  - **spe.names.fr** a vector of strings of character with the species names in french
  - **Spatial** an object of the class SpatialPolygons of sp, containing the map

**Source**

References


Examples

data(avijons)
w1 <- dudi.coa(avijons$fau, scannf = FALSE)$li
area.plot(avijons$area, center = avijons$xy, val = w1[, 1], clab = 0.75,
   sub = "CA Axis 1", csub = 3)

## Not run:
data(avijons)
if(!adegraphicsLoaded()) {
  if(requireNamespace("pixmap", quietly = TRUE)) {
    pnm.eau <- pixmap:::read.pnm(system.file("pictures/avijonseau.pnm", package = "ade4"))
    pnm.rou <- pixmap:::read.pnm(system.file("pictures/avijonsrou.pnm", package = "ade4"))
    pnm.veg <- pixmap:::read.pnm(system.file("pictures/avijonsveg.pnm", package = "ade4"))
    pnm.vil <- pixmap:::read.pnm(system.file("pictures/avijonsvil.pnm", package = "ade4"))
    jons.coa <- dudi.coa(avijons$fau, scan = FALSE, nf = 4)
    par(mfcol = c(3, 2))
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.rou, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+ROADS", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.veg, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+TREES", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.eau, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+WATER", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.rou, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+ROADS", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.veg, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+TREES", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.eau, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+WATER", csub = 3)
    par(mfrow = c(1, 1))
  }
  if(requireNamespace("spdep", quietly = TRUE) &
     requireNamespace("pixmap", quietly = TRUE)) {
    link1 <- area2link(avijons$area)
    lw1 <- apply(link1, 1, function(x) x[x > 0])
    neig1 <- neig(mat01 = 1*(link1 > 0))
    nb1 <- neig2nb(neig1)
    listw1 <- spdep:::nb2listw(nb1, lw1)
    jons.ms <- multispati(jons.coa, listw1, scan = FALSE, nfp = 3, nfn = 2)
    summary(jons.ms)
    par(mfrow = c(2, 2))
    barplot(jons.coa$eig)
    barplot(jons.ms$eig)
    s.corcircle(jons.ms$as)
  }
}

if(!adegraphicsLoaded()) {
  if(requireNamespace("pixmap", quietly = TRUE)) {
    pnm.eau <- pixmap::read.pnm(system.file("pictures/avijonseau.pnm", package = "ade4"))
    pnm.rou <- pixmap::read.pnm(system.file("pictures/avijonsrou.pnm", package = "ade4"))
    pnm.veg <- pixmap::read.pnm(system.file("pictures/avijonsveg.pnm", package = "ade4"))
    pnm.vil <- pixmap::read.pnm(system.file("pictures/avijonsvil.pnm", package = "ade4"))
    jons.coa <- dudi.coa(avijons$fau, scan = FALSE, nf = 4)
    par(mfcol = c(3, 2))
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.rou, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+ROADS", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.veg, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+TREES", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 1], pixmap = pnm.eau, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F1+WATER", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.rou, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+ROADS", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.veg, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+TREES", csub = 3)
    s.value(avijons$xy, jons.coa$li[, 2], pixmap = pnm.eau, inclu = FALSE,
      grid = FALSE, addax = FALSE, cleg = 0, sub = "F2+WATER", csub = 3)
    par(mfrow = c(1, 1))
  }
  if(requireNamespace("spdep", quietly = TRUE) &
     requireNamespace("pixmap", quietly = TRUE)) {
    link1 <- area2link(avijons$area)
    lw1 <- apply(link1, 1, function(x) x[x > 0])
    neig1 <- neig(mat01 = 1*(link1 > 0))
    nb1 <- neig2nb(neig1)
    listw1 <- spdep:::nb2listw(nb1, lw1)
    jons.ms <- multispati(jons.coa, listw1, scan = FALSE, nfp = 3, nfn = 2)
    summary(jons.ms)
    par(mfrow = c(2, 2))
    barplot(jons.coa$eig)
    barplot(jons.ms$eig)
    s.corcircle(jons.ms$as)
}

avimedi

Fauna Table for Constrained Ordinations

Description

avimedi is a list containing the information about 302 sites:
frequencies of 51 bird species; two factors (habitats and Mediterranean origin).

Usage

data(avimedi)

Format

This list contains the following objects:

fau is a data frame 302 sites - 51 bird species.

plan is a data frame 302 sites - 2 factors: reg with two levels Provence (Pr, South of France) and Corsica (Co); str with six levels describing the vegetation from a very low matorral (1) up to a mature forest of holm oaks (6).

nomesp is a vector 51 latin names.

Source

Examples

```r
## Not run:
data(aviurba)
coa1 <- dudi.coa(aviurba$fau, scan = FALSE, nf = 3)
bet1 <- bca(coa1, avurba$plan$str, scan = FALSE)
wit1 <- wca(coa1, avurba$plan$reg, scan=FALSE)
pcaiv1 <- pcaiv(coa1, avurba$plan, scan = FALSE)

if(adegraphicsLoaded()) {
  g1 <- s.class(coa1$li, avurba$plan$str:avurba$plan$reg,
                psub.text = "Correspondences Analysis", plot = FALSE)
  g2 <- s.class(bet1$ls, avurba$plan$str, psub.text = "Between Analysis", plot = FALSE)
  g3 <- s.class(wit1$li, avurba$plan$str, psub.text = "Within Analysis", plot = FALSE)

  g41 <- s.match(pcaiv1$li, pcaiv1$ls, plabels.cex = 0,
                psub.text = "Canonical Correspondences Analysis", plot = FALSE)
  g42 <- s.class(pcaiv1$li, avurba$plan$str:avurba$plan$reg, plot = FALSE)
  g4 <- superpose(g41, g42, plot = FALSE)

  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2,2))
  s.class(coa1$li, avurba$plan$str:avurba$plan$reg,
          sub = "Correspondences Analysis")
  s.class(bet1$ls, avurba$plan$str,
          sub = "Between Analysis")
  s.class(wit1$li, avurba$plan$str,
          sub = "Within Analysis")
  s.match(pcaiv1$li, pcaiv1$ls, clab = 0,
          sub = "Canonical Correspondences Analysis")
  s.class(pcaiv1$li, avurba$plan$str:avurba$plan$reg,
          add.plot = TRUE)
  par(mfrow=c(1,1))
}

## End(Not run)
```

### aviurba

#### Ecological Tables Triplet

**Description**

This data set is a list of information about 51 sites: bird species and environmental variables. A data frame contains biological traits for each species.

**Usage**

```r
data(aviurba)
```
**Format**

This list contains the following objects:

- **fau** is a data frame 51 sites 40 bird species.
- **mil** is a data frame 51 sites 11 environmental variables (see details).
- **traits** is a data frame 40 species 4 biological traits (see details).
- **species.names.fr** is a vector of the species names in french.
- **species.names.la** is a vector of the species names in latin.
- **species.family** is a factor : the species families.

**Details**

**aviurba$mil** contains for each site, 11 habitat attributes describing the degree of urbanization. The presence or absence of farms or villages, small buildings, high buildings, industry, fields, grassland, scrubby areas, deciduous woods, coniferous woods, noisy area are noticed. At least, the vegetation cover (variable 11) is a factor with 8 levels from a minimum cover (R5) up to a maximum (R100).

**aviurba$traits** contains four factors : feeding habit (insectivor, granivore, omnivore), feeding stratum (ground, aerial, foliage and scrub), breeding stratum (ground, building, scrub, foliage) and migration strategy (resident, migrant).

**Source**


**Examples**

```r
data(aviurba)
a1 <- dudi.coa(aviurba$fau, scan = FALSE, nf=4)
a2 <- dudi.acm(aviurba$mil, row.w = a1$lw, scan = FALSE, nf = 4)
plot(coinertia(a1, a2, scan = FALSE))
```

---

**bacteria**

*Genomes of 43 Bacteria*

**Description**

*bacteria* is a list containing 43 species and genomic informations : codons, amino acid and bases.

**Usage**

```r
data(bacteria)
```
**Format**

This list contains the following objects:

- **code** is a factor with the amino acid names for each codon.
- **espcodon** is a data frame 43 species 64 codons.
- **espaa** is a data frame 43 species 21 amino acid.
- **espbase** is a data frame 43 species 4 bases.

**Source**

Data prepared by J. Lobry <Jean.Lobry@univ-lyon1.fr> starting from [https://www.jcvi.org/](https://www.jcvi.org/).

**Examples**

```r
data(bacteria)
names(bacteria$espcodon)
names(bacteria$espaa)
names(bacteria$espbase)
sum(bacteria$espcodon) # 22,619,749 codons

if(adegraphicsLoaded()) {
g <- scatter(dudi.coa(bacteria$espcodon, scann = FALSE),
posi = "bottomleft")
} else {
satter(dudi.coa(bacteria$espcodon, scann = FALSE),
posi = "bottom")
}
```

---

**banque**

<table>
<thead>
<tr>
<th>Table of Factors</th>
</tr>
</thead>
</table>

**Description**

banque gives the results of a bank survey onto 810 customers.

**Usage**

data(banque)

**Format**

This data frame contains the following columns:

1. csp: "Socio-professional categories" a factor with levels
   - agric Farmers
   - artis Craftsmen, Shopkeepers, Company directors
   - cadsu Executives and higher intellectual professions
2. duree: "Time relations with the customer" a factor with levels
   • dm2 <2 years
   • d24 [2 years, 4 years]
   • d48 [4 years, 8 years]
   • d812 [8 years, 12 years]
   • dp12 >= 12 years
3. oppo: "Stopped a check?" a factor with levels
   • non no
   • oui yes
4. age: "Customer’s age" a factor with levels
   • ai25 [18 years, 25 years]
   • ai35 [25 years, 35 years]
   • ai45 [35 years, 45 years]
   • ai55 [45 years, 55 years]
   • ai75 [55 years, 75 years]
5. sexe: "Customer’s gender" a factor with levels
   • hom Male
   • fem Female
6. interdit: "No checkbook allowed" a factor with levels
   • non no
   • oui yes
7. cableue: "Possess a bank card?" a factor with levels
   • non no
   • oui yes
8. assurvi: "Contrat of life insurance?" a factor with levels
   • non no
   • oui yes
9. soldevu: "Balance of the current accounts" a factor with levels
   • p4 credit balance > 20000
   • p3 credit balance 12000-20000
   • p2 credit balance 4000-120000
   • p1 credit balance >0-4000
   • n1 debit balance 0-4000
10. eparlog: "Savings and loan association account amount" a factor with levels
   - for > 20000
   - fai >0 and <20000
   - nul nulle

11. eparliv: "Savings bank amount" a factor with levels
    - for > 20000
    - fai >0 and <20000
    - nul nulle

12. credhab: "Home loan owner" a factor with levels
    - non no
    - oui yes

13. credcon: "Consumer credit amount" a factor with levels
    - nul none
    - fai >0 and <20000
    - for > 20000

14. versesp: "Check deposits" a factor with levels
    - oui yes
    - non no

15. retresp: "Cash withdrawals" a factor with levels
    - fai < 2000
    - moy 2000-5000
    - for > 5000

16. remiche: "Endorsed checks amount" a factor with levels
    - for >10000
    - moy 10000-5000
    - fai 1-5000
    - nul none

17. preltre: "Treasury Department tax deductions" a factor with levels
    - nul none
    - fai <1000
    - moy >1000

18. prelfin: "Financial institution deductions" a factor with levels
    - nul none
    - fai <1000
    - moy >1000

19. viredeb: "Debit transfer amount" a factor with levels
    - nul none
    - fai <2500
• moy 2500-5000
• for >5000

20. virecre: "Credit transfer amount" a factor with levels
• for >10000
• moy 10000-5000
• fai <5000
• nul aucun

21. porttit: "Securities portfolio estimations" a factor with levels
• nul none
• fai < 20000
• moy 20000-100000
• for >100000

Source
anonymous

Examples

data(banque)
banque.acm <- dudi.acm(banque, scannf = FALSE, nf = 3)
apply(banque.acm$cr, 2, mean)
banke.acm$eig[1:banque.acm$nf] # the same thing

if(adeographicsLoaded()) {
  g <- s.arrow(banque.acm$c1, plabels.cex = 0.75)
} else {
  s.arrow(banque.acm$c1, clab = 0.75)
}

baran95

African Estuary Fishes

Description
This data set is a list containing relations between sites and fish species linked to dates.

Usage

data(baran95)
Format

This list contains the following objects:

- **fau** is a data frame 95 seinings and 33 fish species.
- **plan** is a data frame 2 factors: date and site. The date has 6 levels (April 1993, June 1993, August 1993, October 1993, December 1993 and February 1994) and the sites are defined by 4 distances to the Atlantic Ocean (km03, km17, km33 and km46).
- **species.names** is a vector of species Latin names.

Source


References


Examples

data(baran95)
w <- dudi.pca(log(baran95$fau + 1), scal = FALSE, scann = FALSE, nf = 3)
w1 <- wca(w, baran95$plan$date, scann = FALSE)
fatala <- ktab.within(w1)
stat1 <- statis(fatala, scan = FALSE, nf = 3)
mfa1 <- mfa(fatala, scan = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  g1 <- s.class(stat1$C.Co, baran95$plan$site, facets = baran95$plan$date, pellipses.axes.draw = FALSE, ppoints.cex = 0.5, plot = FALSE)
n1 <- length(g1$ADEglist)
g2 <- ADEgS(lapply(1:n1, function(i) s.label(stat1$C.Co, plabels.cex = 0, ppoints.cex = 0.5, plot = FALSE)), positions = g1$positions, plot = FALSE)
G1 <- superpose(g2, g1, plot = TRUE)
G2 <- kplot(stat1, arrow = FALSE, traject = FALSE, class = baran95$plan$site, col.plabels.cex = 0, ppoints.cex = 0.5)

g3 <- s.class(mfa1$co, baran95$plan$site, facets = baran95$plan$date, pellipses.axes.draw = FALSE, ppoints.cex = 0.5, plot = FALSE)
n2 <- length(g3$ADEglist)
g4 <- ADEgS(lapply(1:n2, function(i) s.label(mfa1$co, plabels.cex = 0, ppoints.cex = 0.5, plot = FALSE)), positions = g3$positions, plot = FALSE)
G3 <- superpose(g4, g3, plot = TRUE)
} else {
  par(mfrow = c(3, 2))
w2 <- split(stat1$C.Co, baran95$plan$date)
w3 <- split(baran95$plan$site, baran95$plan$date)
bca

### Between-Class Analysis

**Description**

Performs a particular case of a Principal Component Analysis with respect to Instrumental Variables (pcaiv), in which there is only a single factor as explanatory variable.

**Usage**

```r
## S3 method for class 'dudi'
bcax, fac, scannf = TRUE, nf = 2, ...)
```

**Arguments**

- `x` a duality diagram, object of class `dudi` from one of the functions `dudi.coa, dudi.pca`,...
- `fac` a factor partitioning the rows of `dudi$tab` in classes
- `scannf` a logical value indicating whether the eigenvalues barplot should be displayed
- `nf` if `scannf` FALSE, a numeric value indicating the number of kept axes
- `...` further arguments passed to or from other methods

**Value**

Returns a list of class `dudi`, subclass `between` containing

- `tab` a data frame class-variables containing the means per class for each variable
- `cw` a numeric vector of the column weigths
- `lw` a numeric vector of the class weigths
bca

eig
rank
nf
c1
l1
c0
l1
call
ratio
ls
as

Note
To avoid conflict names with the base:::within function, the function within is now deprecated and removed. To be consistent, the between function is also deprecated and is replaced by the method bca.dudi of the new generic bca function.

Author(s)
Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References

Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
pca2 <- dudi.pca(meaudret$spe, scal = FALSE, scan = FALSE, nf = 4)
bet1 <- bca(pca1, meaudret$design$site, scan = FALSE, nf = 2)
bet2 <- bca(pca2, meaudret$design$site, scan = FALSE, nf = 2)

if(adegraphicsLoaded()) {
  g1 <- s.class(pca1$l1, meaudret$design$site, psub.text = "Principal Component Analysis (env)", plot = FALSE)
g2 <- s.class(pca2$l1, meaudret$design$site, psub.text = "Principal Component Analysis (spe)", plot = FALSE)
g3 <- s.class(bet1$ls, meaudret$design$site, psub.text = "Between sites PCA (env)", plot = FALSE)
g4 <- s.class(bet2$ls, meaudret$design$site, psub.text = "Between sites PCA (spe)", plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
s.class(pca1$li, meaudret$design$site, sub = "Principal Component Analysis (env)", csub = 1.75)
s.class(pca2$li, meaudret$design$site, sub = "Principal Component Analysis (spe)", csub = 1.75)
s.class(bet1$ls, meaudret$design$site, sub = "Between sites PCA (env)", csub = 1.75)
s.class(bet2$ls, meaudret$design$site, sub = "Between sites PCA (spe)", csub = 1.75)
par(mfrow = c(1, 1))
}

coib <- coinertia(bet1, bet2, scann = FALSE)
plot(coib)

---

**bcaw.oinertia**

**Between-class coinertia analysis**

**Description**

Performs a between-class analysis after a coinertia analysis

**Usage**

```r
## S3 method for class 'coinertia'
bca(x, fac, scannf = TRUE, nf = 2, ...)
```

**Arguments**

- `x`: a coinertia analysis (object of class `coinertia`) obtained by the function `coinertia`
- `fac`: a factor partitioning the rows in classes
- `scannf`: a logical value indicating whether the eigenvalues barplot should be displayed
- `nf`: if `scannf` FALSE, an integer indicating the number of kept axes
- `...`: further arguments passed to or from other methods

**Details**

This analysis is equivalent to do a between-class analysis on each initial dudi, and a coinertia analysis on the two between analyses. This function returns additional outputs for the interpretation.

**Value**

An object of the class `betcoi`. Outputs are described by the `print` function

**Note**

To avoid conflict names with the `base::within` function, the function `within` is now deprecated and removed. To be consistent, the `betweencoinertia` function is also deprecated and is replaced by the method `bca.coinertia` of the new generic `bca` function.

**Author(s)**

Stéphane Dray `<stephane.dray@univ-lyon1.fr>` and Jean Thioulouse `<jean.thioulouse@univ-lyon1.fr>`
References

See Also
coinertia, bca

Examples
data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
pca2 <- dudi.pca(meaudret$spe, scal = FALSE, scan = FALSE, nf = 4)

bet1 <- bca(pca1, meaudret$design$site, scan = FALSE, nf = 2)
bet2 <- bca(pca2, meaudret$design$site, scan = FALSE, nf = 2)
coib <- coinertia(bet1, bet2, scannf = FALSE)

coi <- coinertia(pca1, pca2, scannf = FALSE)
coi.b <- bca(coi, meaudret$design$site, scannf = FALSE)
## coib and coi.b are equivalent

plot(coi.b)

bcar1q \hspace{1cm} Between-Class RLQ analysis

Description
Performs a particular RLQ analysis where a partition of sites (rows of R) is taken into account. The between-class RLQ analysis search for linear combinations of traits and environmental variables maximizing the covariances between the traits and the average environmental conditions of classes.

Usage
## S3 method for class 'rlq'
bca(x, fac, scannf = TRUE, nf = 2, ...)
## S3 method for class 'betrlq'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'betrlq'
print(x, ...)

Arguments

x \hspace{1cm} an object of class rlq (created by the rlq function) for the bca.r1q function. An object of class betrlq for the print and plot functions

fac \hspace{1cm} a factor partitioning the rows of R

scannf \hspace{1cm} a logical value indicating whether the eigenvalues bar plot should be displayed
Between-Class Analysis

**Description**

Outputs and graphical representations of the results of a between-class analysis.

**nf**  
if scannf FALSE, an integer indicating the number of kept axes

**xax**  
the column number for the x-axis

**yax**  
the column number for the y-axis

...  
further arguments passed to or from other methods

**Value**

The `bca.rlq` function returns an object of class 'betrlq' (sub-class of 'dudi'). See the outputs of the `print` function for more details.

**Author(s)**

Stéphane Dray <stephane.dray@univ-lyon1.fr>

**References**


**See Also**

`rlq`, `bca`, `wca.rlq`

**Examples**

```r
data(piosphere)
afcL <- dudi.coa(log(piosphere$veg + 1), scannf = FALSE)
acpR <- dudi.pca(piosphere$env, scannf = FALSE, row.w = afcL$lw)
acpQ <- dudi.hillsmith(piosphere$traits, scannf = FALSE, row.w = afcL$cw)
rlq1 <- rlq(acpR, afcL, acpQ, scannf = FALSE)
brlq1 <- bca(rlq1, fac = piosphere$habitat, scannf = FALSE)
brlq1
plot(brlq1)
```
Usage

```r
## S3 method for class 'between'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'between'
print(x, ...)
## S3 method for class 'betcoi'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'betcoi'
print(x, ...)
## S3 method for class 'between'
summary(object, ...)
```

Arguments

- `x, object` an object of class `between` or `betcoi`
- `xax, yax` the column index of the x-axis and the y-axis
- `...` further arguments passed to or from other methods

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also

- `bca.dudi`, `bca.coinertia`

Examples

```r
data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
pca2 <- dudi.pca(meaudret$spe, scal = FALSE, scan = FALSE, nf = 4)
bet1 <- bca(pca1, meaudret$design$site, scan = FALSE, nf = 2)
bet2 <- bca(pca2, meaudret$design$site, scan = FALSE, nf = 2)

if(adegraphicsLoaded()) {
  g1 <- s.class(pca1$li, meaudret$design$site, psub.text = "Principal Component Analysis (env)",
                plot = FALSE)
  g2 <- s.class(pca2$li, meaudret$design$site, psub.text = "Principal Component Analysis (spe)",
                plot = FALSE)
  g3 <- s.class(bet1$ls, meaudret$design$site, psub.text = "Between sitesPCA (env)",
                plot = FALSE)
```
g4 <- s.class(bet2$ls, meaudret$design$site, psub.text = "Between sites PCA (spe)", plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))

} else {
  par(mfrow = c(2, 2))
s.class(pca1$li, meaudret$design$site, sub = "Principal Component Analysis (env)", csub = 1.75)
s.class(pca2$li, meaudret$design$site, sub = "Principal Component Analysis (spe)", csub = 1.75)
s.class(bet1$ls, meaudret$design$site, sub = "Between sites PCA (env)", csub = 1.75)
s.class(bet2$ls, meaudret$design$site, sub = "Between sites PCA (spe)", csub = 1.75)
  par(mfrow = c(1,1))
}
coib <- coinertia(bet1, bet2, scann = FALSE)
plot(coib)

---

bf88  

Cubic Ecological Data

Description

bf88 is a list of 6 data frames corresponding to 6 stages of vegetation. Each data frame gives some bird species informations for 4 counties.

Usage

data(bf88)

Format

A list of six data frames with 79 rows (bird species) and 4 columns (counties). The 6 arrays (S1 to S6) are the 6 stages of vegetation. The attribut 'nomesp' of this list is a vector of species French names.

Source


Examples

data(bf88)
fou1 <- foucart(bf88, scann = FALSE, nf = 3)
fou1

if(adegraphicsLoaded()) {
  g1 <- scatter(fou1, plot = FALSE)
g2 <- s.traject(fou1$Tco, fou1$TC[, 1], plines.lty = 1:length(levels(fou1$TC[, 1])), plot = FALSE)
g3 <- s.traject(fou1$Tco, fou1$TC[, 2], plines.lty = 1:length(levels(fou1$TC[, 2])), plot = FALSE)
bicenter.wt

**Double Weighted Centring**

**Description**

This function creates a doubly centred matrix.

**Usage**

```r
bicenter.wt(X, row.wt = rep(1, nrow(X)), col.wt = rep(1, ncol(X)))
```

**Arguments**

- **X**: a matrix with n rows and p columns
- **row.wt**: a vector of positive or null weights of length n
- **col.wt**: a vector of positive or null weights of length p

**Value**

returns a doubly centred matrix

**Author(s)**

Daniel Chessel

**Examples**

```r
w <- matrix(1:6, 3, 2)
bicenter.wt(w, c(0.2, 0.6, 0.2), c(0.3, 0.7))

w <- matrix(1:20, 5, 4)
sum(bicenter.wt(w, runif(5), runif(4))^2)
```
**Description**

The `bordeaux` data frame gives the opinions of 200 judges in a blind tasting of five different types of claret (red wine from the Bordeaux area in the south western parts of France).

**Usage**

```r
data(bordeaux)
```

**Format**

This data frame has 5 rows (the wines) and 4 columns (the judgements) divided in excellent, good, mediocre and boring.

**Source**


**Examples**

```r
data(bordeaux)
bordeaux
score(dudi.coa(bordeaux, scan = FALSE))
```

---

**Description**

This data set gives ecological and biological characteristics of 131 species of aquatic insects.

**Usage**

```r
data(bsetal97)
```
Format

bsetal97 is a list of 8 components.

species.names is a vector of the names of aquatic insects.
taxo is a data frame containing the taxonomy of species: genus, family and order.
biol is a data frame containing 10 biological traits for a total of 41 modalities.
biol.blo is a vector of the numbers of items for each biological trait.
biol.blo.names is a vector of the names of the biological traits.
ecol is a data frame with 7 ecological traits for a total of 34 modalities.
ecol.blo is a vector of the numbers of items for each ecological trait.
ecol.blo.names is a vector of the names of the ecological traits.

Details

The 10 variables of the data frame bsetal97$biol are called in bsetal97$biol.blo.names and the number of modalities per variable given in bsetal97$biol.blo. The variables are: female size - the body length from the front of the head to the end of the abdomen (7 length modalities), egg length - the egg size (6 modalities), egg number - count of eggs actually oviposited, generations per year (3 modalities: ≤1, 2, > 2), oviposition period - the length of time during which oviposition occurred (3 modalities: ≤2 months, between 2 and 5 months, >5 months), incubation time - the time between oviposition and hatching of the larvae (3 modalities: ≤4 weeks, between 4 and 12 weeks, >12 weeks), egg shape (1-spherical, 2-oval, 3-cylindrical), egg attachment - physiological feature of the egg and of the female (4 modalities), clutch structure (1-single eggs, 2-grouped eggs, 3-egg masses), clutch number (3 modalities: 1, 2, >2).

The 7 variables of the data frame bsetal97$ecol are called in bsetal97$ecol.blo.names and the number of modalities per variable given in bsetal97$ecol.blo. The variables are: oviposition site - position relative to the water (7 modalities), substratum type for eggs - the substratum to which the eggs are definitely attached (6 modalities), egg deposition - the position of the eggs during the oviposition process (4 modalities), gross habitat - the general habitat use of the species such as temporary waters or estuaries (8 modalities), saturation variance - the exposure of eggs to the risk of dessication (2 modalities), time of day (1-morning, 2-day, 3-evening, 4-night), season - time of the year (1-Spring, 2-Summer, 3-Autumn).

Source


References

Examples

data(bsetal97)
X <- prep.fuzzy.var(bsetal97$biol, bsetal97$biol.blo)
Y <- prep.fuzzy.var(bsetal97$ecol, bsetal97$ecol.blo)
plot(coinertia(dudi.fca(X, scan = FALSE),
              dudi.fca(Y, scan = FALSE), scan = FALSE))

Description

This data set contains information about Buech basin characteristics.

Usage

data(buech)

Format

buech is a list with the following components:

- **tab1** a data frame with 10 environmental variables collected on 31 sites in June (1984)
- **tab2** a data frame with 10 environmental variables collected on 31 sites in September (1984)
- **xy** a data frame with the coordinates of the sites
- **neig** an object of class neig
- **contour** a data frame for background map
- **nb** the neighbouring graph between sites, object of the class nb
- **Spatial** an object of the class SpatialPolygons of sp, containing the map

Details

Variables of `buech$tab1` and `buech$tab2` are the following ones:

- pH
- Conductivity (µ S/cm)
- Carbonate (water hardness (mg/l CaCO3))
- Hardness (total water hardness (mg/l CaCO3))
- Bicarbonate (alkalinity (mg/l HCO3-))
- Chloride (alkalinity (mg/l Cl-))
- Suspens (particles in suspension (mg/l))
- Organic (organic particles (mg/l))
- Nitrate (nitrate rate (mg/l NO3-))
- Ammonia (ammoniac rate (mg/l NH4-))

Source


data(buech)
if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g1 <- s.label(buech$xy, Sp = buech$Spatial, nb = buech$nb,
                  pSp.col = "transparent", plot = FALSE)
    g2 <- s.value(buech$xy, buech$tab2$Suspens - buech$tab1$Suspens,
                  Sp = buech$Spatial, nb = buech$nb, pSp.col = "transparent", plot = FALSE)
    G <- cbindADEg(g1, g2, plot = TRUE)
  }
} else {
  par(mfrow = c(1,2))
  s.label(buech$xy, contour = buech$contour, neig = buech$neig)
  s.value(buech$xy, buech$tab2$Suspens - buech$tab1$Suspens,
           contour = buech$contour, neig = buech$neig, csi = 3)
  par(mfrow = c(1,1))
}

butterfly is a list with the following components:

- xy: a data frame with the two coordinates of the 16 Euphydryas editha butterfly colonies
- envir: a environmental data frame of 16 sites - 4 variables
- genet: a genetics data frame of 16 sites - 6 allele frequencies
- contour: a data frame for background map (California map)
- Spatial: an object of the class SpatialPolygons of sp, containing the map


Examples

data(butterfly)

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g1 <- s.label(butterfly$xy, Sp = butterfly$Spatial, pSp.col = "white",
                  porigin.include = FALSE, plot = FALSE)
    g2 <- table.value(dist(butterfly$xy), plot = FALSE)
    g3 <- s.value(butterfly$xy, dudi.pca(butterfly$envir, scan = FALSE)$li[, 1],
                  Sp = butterfly$Spatial, pori.inc = FALSE, pSp.col = "transparent", ppoints.cex = 2,
                  plot = FALSE)
    ## mt <- mantel.randtest(dist(butterfly$xy), dist(butterfly$gen), 99)
    G <- ADEgS(list(g1, g2, g3), layout = c(2, 2), plot = TRUE)
  }
} else {
  par(mfrow = c(2, 2))
  s.label(butterfly$xy, contour = butterfly$contour, inc = FALSE)
  table.dist(dist(butterfly$xy), labels = row.names(butterfly$xy)) # depends of mva
  s.value(butterfly$xy, dudi.pca(butterfly$envir, scan = FALSE)$li[,1],
          contour = butterfly$contour, inc = FALSE, csi = 3)
  plot(mantel.randtest(dist(butterfly$xy), dist(butterfly$gen), 99),
       main = "genetic/spatial")
  par(mfrow = c(1,1))
}

bwca.dpcoa

Between- and within-class double principal coordinate analysis

Description

These functions allow to study the variations in diversity among communities (as in dpcoa) taking
into account a partition in classes

Usage

bwca.dpcoa(x, fac, cofac, scannf = TRUE, nf = 2, ...)
## S3 method for class 'dpcoa'
bca(x, fac, scannf = TRUE, nf = 2, ...)
## S3 method for class 'dpcoa'
wca(x, fac, scannf = TRUE, nf = 2, ...)
## S3 method for class 'betwit'
randtest(xtest, nrepet = 999, ...)
## S3 method for class 'betwit'
summary(object, ...)
## S3 method for class 'witdpcoa'
print(x, ...)
Arguments

- `x`: an object of class `dpcoa`
- `fac`: a factor partitioning the collections in classes
- `scannf`: a logical value indicating whether the eigenvalues barplot should be displayed
- `nf`: if `scannf` FALSE, a numeric value indicating the number of kept axes
- `...`: further arguments passed to or from other methods
- `cofac`: a cofactor partitioning the collections in classes used as a covariable
- `nrepet`: the number of permutations
- `xtest, object`: an object of class `betwit` created by a call to the function `bwca.dpcoa`

Value

Objects of class `betdpcoa`, `witdpcoa` or `betwit`

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also

dpcoa

Examples

```r
## Not run:
## First example of Dray et al (2015) paper

con <- url("https://pbil.univ-lyon1.fr/datasets/dray/MER2014/soilmicrob.rda")
load(con)
close(con)

## Partial CCA
coa <- dudi.coa(soilmicrob$OTU, scannf = FALSE)
wcoa <- wca(coa, soilmicrob$env$ph, scannf = FALSE)
wbcoa <- bca(wcoa, soilmicrob$env$VegType, scannf = FALSE)

## Classical DPCoA
dp <- dpcoa(soilmicrob$OTU, soilmicrob$dphy, RaoDecomp = FALSE, scannf = FALSE)

## Between DPCoA (focus on the effect of vegetation type)
```
cailliez

Transformation to make Euclidean a distance matrix

Description
This function computes the smallest positive constant that makes Euclidean a distance matrix and applies it.

Usage
cailliez(distmat, print = FALSE, tol = 1e-07, cor.zero = TRUE)

Arguments
distmat an object of class dist
print if TRUE, prints the eigenvalues of the matrix
tol a tolerance threshold for zero
cor.zero if TRUE, zero distances are not modified

Value
an object of class dist containing a Euclidean distance matrix.

Author(s)
Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
References


Examples
data(capitales)
d0 <- capitales$dist
is.euclid(d0)  # FALSE
d1 <- cailliez(d0, TRUE)
  # Cailliez constant = 2429.87867
is.euclid(d1)  # TRUE
plot(d0, d1)
abline(lm(unclass(d1)~unclass(d0)))
print(coefficients(lm(unclass(d1)~unclass(d0))), dig = 8)  # d1 = d + Cte
is.euclid(d0 + 2428)  # FALSE
is.euclid(d0 + 2430)  # TRUE the smallest constant

<table>
<thead>
<tr>
<th>capitales</th>
<th>Road Distances</th>
</tr>
</thead>
</table>

Description
This data set gives the road distances between 15 European capitals and their coordinates.

Usage
data(capitales)

Format
capitales is a list with the following components:
xxy a data frame containing the coordinates of capitals
area a data frame containing three variables, designed to be used in area.plot function
logo a list of pixmap objects, each one symbolizing a capital
Spatial an object of the class SpatialPolygons of sp, containing the map
dist a dist object the road distances between 15 European capitals
Examples

```r
data(capitales)
attr(capitales$dist, "Labels")
index <- pmatch(tolower(attr(capitales$dist, "Labels")), names(capitales$logo))
w1 <- capitales$area

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g1 <- s.label(capitales$xy, lab = rownames(capitales$xy), porigin.include = FALSE,
                  plot = FALSE)
    g2 <- s.logo(capitales$xy[sort(rownames(capitales$xy)), ], capitales$logo,
                 Sp = capitales$Spatial, pbackground.col = "lightblue", pSp.col = "white",
                 pgrid.draw = FALSE, plot = FALSE)
    g3 <- table.value(capitales$dist, ptable.margin = list(b = 5, l = 5, t = 15, r = 15),
                    ptable.x.tck = 3, ptable.y.tck = 3, plot = FALSE)
    g4 <- s.logo(pcoscaled(lingoes(capitales$dist)), capitales$logo[index], plot = FALSE)
    G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
  }
  else {
    if(requireNamespace("pixmap", quietly = TRUE)) {
      par(mfrow = c(2, 2))
      s.label(capitales$xy, lab = attr(capitales$dist, "Labels"), include.origin = FALSE)
      area.plot(w1)
      rect(min(w1$x), min(w1$y), max(w1$x), max(w1$y), col = "lightblue")
      invisible(lapply(split(w1, w1$id), function(x) polygon(x[, -1], col = "white")))
      s.logo(capitales$xy, capitales$logo, klogo = index, add.plot = TRUE,
             include.origin = FALSE, clogo = 0.5) # depends on pixmap
      table.dist(capitales$dist, lab = attr(capitales$dist, "Labels")) # depends on mva
      s.logo(pcoscaled(lingoes(capitales$dist)), capitales$logo, klogo = index, clogo = 0.5)
      # depends on pixmap
      par(mfrow = c(1, 1))
    }
  }
}
```

Description

This data set describes the phylogeny of carnivora as reported by Diniz-Filho et al. (1998). It also gives the body mass of these 19 species.

Usage

```r
data(carni19)
```
carni70

**Format**

carni19 is a list containing the 2 following objects:

- **tre** is a character string giving the phylogenetic tree in Newick format.
- **bm** is a numeric vector which values correspond to the body mass of the 19 species (log scale).

**Source**


**Examples**

```r
data(carni19)
carni19.phy <- newick2phylog(carni19$tre)
par(mfrow = c(1,2))
symbols.phylog(carni19.phy,carni19$bm-mean(carni19$bm))
dotchart.phylog(carni19.phy, carni19$bm, clabel.l=0.75)
par(mfrow = c(1,1))
```

---

carni70

**Phylogeny and quantitative traits of carnivora**

**Description**

This data set describes the phylogeny of 70 carnivora as reported by Diniz-Filho and Torres (2002). It also gives the geographic range size and body size corresponding to these 70 species.

**Usage**

```r
data(carni70)
```

**Format**

carni70 is a list containing the 2 following objects:

- **tre** is a character string giving the phylogenetic tree in Newick format. Branch lengths are expressed as divergence times (millions of years)
- **tab** is a data frame with 70 species and two traits: size (body size (kg)) ; range (geographic range size (km)).

**Source**

Examples

```r
## Not run:
if (requireNamespace("adephylo", quietly = TRUE) & requireNamespace("ape", quietly = TRUE)) {
  data(carni70)
  carni70.phy <- newick2phylog(carni70$tre)
  plot(carni70.phy)

  size <- scalewt(log(carni70$tab))[,1]
  names(size) <- row.names(carni70$tab)
  symbols.phylog(carni70.phy, size)

  tre <- ape::read.tree(text = carni70$tre)
  adephylo::orthogram(size, tre = tre)

  yrange <- scalewt(carni70$tab[,2])
  names(yrange) <- row.names(carni70$tab)
  symbols.phylog(carni70.phy, yrange)
  adephylo::orthogram(as.vector(yrange), tre = tre)

  if(adegraphicsLoaded()) {
    g1 <- s.label(cbind.data.frame(size, yrange), plabel.cex = 0)
    g2 <- addhist(g1)
  } else {
    s.hist(cbind.data.frame(size, yrange), clabel = 0)
  }
}
## End(Not run)
```

carniherbi49

### Taxonomy, phylogenies and quantitative traits of carnivora and herbivora

Description

This data set describes the taxonomic and phylogenetic relationships of 49 carnivora and herbivora species as reported by Garland and Janis (1993) and Garland et al. (1993). It also gives seven traits corresponding to these 49 species.

Usage

data(carniherbi49)

Format

carniherbi49 is a list containing the 5 following objects:

- **taxo** is a data frame with 49 species and 2 columns: `fam`, a factor family with 14 levels and `ord`, a factor order with 3 levels.
**tre1** is a character string giving the phylogenetic tree in Newick format as reported by Garland et al. (1993).

**tre2** is a character string giving the phylogenetic tree in Newick format as reported by Garland and Janis (1993).

**tab1** is a data frame with 49 species and 2 traits: 'bodymass' (body mass (kg)) and 'homerange' (home range (km)).

**tab2** is a data frame with 49 species and 5 traits: 'clade' (dietary with two levels Carnivore and Herbivore), 'runningspeed' (maximal sprint running speed (km/h)), 'bodymass' (body mass (kg)), 'hindlength' (hind limb length (cm)) and 'mtfratio' (metatarsal/femur ratio).

**Source**


**Examples**

```r
## Not run:
data(carniherbi49)
par(mfrow=c(1,3))
plot(newick2phylog(carniherbi49$tre1), clabel.leaves = 0,
     f.phylog = 2, sub = "article 1")
plot(newick2phylog(carniherbi49$tre2), clabel.leaves = 0,
     f.phylog = 2, sub = "article 2")
taxo <- as.taxo(carniherbi49$taxo)
plot(taxo2phylog(taxo), clabel.nodes = 1.2, clabel.leaves = 1.2)
par(mfrow = c(1,1))
## End(Not run)
```

---

**casitas**

Entzymatic polymorphism in *Mus musculus*

**Description**

This data set is a data frame with 74 rows (mice) and 15 columns (loci enzymatic polymorphism of the DNA mitochondrial). Each value contains 6 characters coding for two alleles. The missing values are coding by '000000'.

**Usage**

data(casitas)
**Format**

The 74 individuals of casitas belong to 4 groups:

1. 24 mice of the sub-species *Mus musculus domesticus*
2. 11 mice of the sub-species *Mus musculus castaneus*
3. 9 mice of the sub-species *Mus musculus musculus*
4. 30 mice from a population of the lake Casitas (California)

**Source**

Exemple du logiciel GENETIX. Belkhir k. et al. GENETIX, logiciel sous WindowsTM pour la génétique des populations. Laboratoire Génome, Populations, Interactions CNRS UMR 5000, Université de Montpellier II, Montpellier (France).

https://kimura.univ-montp2.fr/genetix/

**References**


**Examples**

```r
data(casitas)
str(casitas)
names(casitas)
```

chatcat  

<table>
<thead>
<tr>
<th>Qualitative Weighted Variables</th>
</tr>
</thead>
</table>

**Description**

This data set gives the age, the fecundity and the number of litters for 26 groups of cats.

**Usage**

```r
data(chatcat)
```

**Format**

chatcat is a list of two objects:

- **tab** is a data frame with 3 factors (age, feco, nport).
- **eff** is a vector of numbers.
Details

One row of tab corresponds to one group of cats.
The value in eff is the number of cats in this group.

Source


Examples

data(chatcat)
summary(chatcat$tab)
w <- acm.disjonctif(chatcat$tab) # Disjonctive table
names(w) <- c(paste("A", 1:5, sep = ""), paste("B", 1:5, sep = ""),
paste("C", 1:2, sep = ""))
w <- t(w*chatcat$num)
w <- data.frame(w)
w # BURT table

chats

<table>
<thead>
<tr>
<th>Pair of Variables</th>
</tr>
</thead>
</table>

Description

This data set is a contingency table of age classes and fecundity classes of cats Felis catus.

Usage

data(chats)

Format

chats is a data frame with 8 rows and 8 columns.
The 8 rows are age classes (age1, ..., age8).
The 8 columns are fecundity classes (f0, f12, f34, ..., fcd).
The values are cats numbers (contingency table).

Source

Examples

data(chats)
chatsw <- as.table(t(chats))
chatscoa <- dudi.coa(data.frame(t(chats)), scann = FALSE)

if(adegraphicsLoaded()) {
  g1 <- table.value(chatsw, ppoints.cex = 1.3, meanX = TRUE, ablineX = TRUE, plabel.cex = 1.5,
                    plot = FALSE)
  g2 <- table.value(chatsw, ppoints.cex = 1.3, meanY = TRUE, ablineY = TRUE, plabel.cex = 1.5,
                    plot = FALSE)
  g3 <- table.value(chatsw, ppoints.cex = 1.3, coordsx = chatscoa$c1[, 1], coordsy = chatscoa$l1[, 1], meanX = TRUE, ablineX = TRUE, plot = FALSE)
  g4 <- table.value(chatsw, ppoints.cex = 1.3, meanY = TRUE, ablineY = TRUE, coordsx = chatscoa$c1[, 1], coordsy = chatscoa$l1[, 1], plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  table.cont(chatsw, abmean.x = TRUE, csi = 2, abline.x = TRUE, clabel.r = 1.5, clabel.c = 1.5)
  table.cont(chatsw, abmean.y = TRUE, csi = 2, abline.y = TRUE, clabel.r = 1.5, clabel.c = 1.5)
  table.cont(chatsw, x = chatscoa$c1[, 1], y = chatscoa$l1[, 1], abmean.x = TRUE, csi = 2,
             abline.x = TRUE, clabel.r = 1.5, clabel.c = 1.5)
  table.cont(chatsw, x = chatscoa$c1[, 1], y = chatscoa$l1[, 1], abmean.y = TRUE, csi = 2,
             abline.y = TRUE, clabel.r = 1.5, clabel.c = 1.5)
  par(mfrow = c(1, 1))
}

---

chazeb  Charolais-Zebus

Description

This data set gives six different weights of 23 charolais and zebu oxen.

Usage

data(chazeb)

Format

chazeb is a list of 2 components.

- **tab** is a data frame with 23 rows and 6 columns.
- **cla** is a factor with two levels "cha" and "zeb".

Source

Examples

```r
data(chazeb)
if(!adegraphicsLoaded())
  plot(discrimin(dudi.pca(chazeb$tab, scan = FALSE),
                chazeb$cla, scan = FALSE))
```

---

**Description**

This data set contains a list of three components: spatial map, allelic profiles and sample sizes.

**Usage**

```r
data(chevaine)
```

**Format**

This data set is a list of three components:

- **tab** a data frame with 27 populations and 9 allelic frequencies (4 locus)
- **coo** a list containing all the elements to build a spatial map
- **eff** a numeric containing the numbers of fish samples per station

**References**


**Examples**

```r
data(chevaine)
names(chevaine)
str(chevaine)
```
Veterinary epidemiological study to assess the risk factors for losses in broiler chickens

Description

This data set contains information about potential risk factors for losses in broiler chickens

Usage

data(chickenk)

Format

A list with 5 components:

- **mortality**: a data frame with 351 observations and 4 variables which describe the losses (dependent dataset Y)
- **FarmStructure**: a data frame with 351 observations and 5 variables which describe the farm structure (explanatory dataset)
- **OnFarmHistory**: a data frame with 351 observations and 4 variables which describe the flock characteristics at placement (explanatory dataset)
- **FlockCharacteristics**: a data frame with 351 observations and 6 variables which describe the flock characteristics during the rearing period (explanatory dataset)
- **CatchingTranspSlaught**: a data frame with 351 observations and 5 variables which describe the transport, lairage conditions, slaughterhouse and inspection features (explanatory dataset)

Source


Examples

data(chickenk)
ktai <- ktab.list.df(chickenk)
**clementines**

**Fruit Production**

**Description**

The clementines is a data set containing the fruit production of 20 clementine trees during 15 years.

**Usage**

```r
data(clementines)
```

**Format**

A data frame with 15 rows and 20 columns

**Source**


**Examples**

```r
data(clementines)

op <- par(no.readonly = TRUE)
par(mfrow = c(5, 4))
par(mar = c(2, 2, 1, 1))
for(i in 1:20) {
  w0 <- 1:15
  plot(w0, clementines[, i], type = "b")
  abline(lm(clementines[, i] ~ w0))
}
par(op)

pca1 <- dudi.pca(clementines, scan = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.corcircle(pca1$co, plab.cex = 0.75)
  g2 <- s1d.barchart(pca1$li[, 1], pld.hori = FALSE)
} else {
  s.corcircle(pca1$co, clab = 0.75)
  barplot(pca1$li[, 1])
}

op <- par(no.readonly = TRUE)
par(mfrow = c(5, 4))
par(mar = c(2, 2, 1, 1))
clem0 <- pca1$tab
croi <- 1:15
```
alter <- c(rep(c(1, -1), 7), 1)
for(i in 1:20) {
    y <- clem0[,i]
    plot(w0, y, type = "b", ylim = c(-2, 2))
    z <- predict(lm(clem0[, i] ~ croi * alter))
    points(w0, z, pch = 20, cex = 2)
    for(j in 1:15)
        segments(j, y[j], j, z[j])
}
par(op)
par(mfrow = c(1, 1))

cnc2003

Frequenting movie theaters in France in 2003

Description

cnc2003 is a data frame with 94 rows (94 departments from continental Metropolitan France) and 12 variables.

Usage

data(cnc2003)

Format

This data frame contains the following variables:

popu  is the population department in million inhabitants.
entr  is the number of movie theater visitors in million.
rece  is the takings from ticket offices.
sean  is the number of proposed shows in thousands.
comm  is the number of equipped communes in movie theaters (units).
etab  is the number of active movie theaters (units).
salle is the number of active screens.
faut is the number of proposed seats.
artes is the number of movie theaters offering "Art and Essay" movies.
multi is the number of active multiplexes.
depart is the name of the department.
reg   is the administrative region of the department.

Source

National Center of Cinematography (CNC), september 2003
The coinertia analysis performs a double inertia analysis of two tables.

**Usage**

```r
coinertia(dudiX, dudiY, scannf = TRUE, nf = 2)
## S3 method for class 'coinertia'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'coinertia'
print(x, ...)
## S3 method for class 'coinertia'
summary(object, ...)
```

**Arguments**

- `dudiX`: a duality diagram providing from one of the functions `dudi.coa`, `dudi.pca`, ...
- `dudiY`: a duality diagram providing from one of the functions `dudi.coa`, `dudi.pca`, ...
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: if `scannf` FALSE, an integer indicating the number of kept axes
- `x`, `object`: an object of class ’coinertia’
- `xax`, `yax`: the numbers of the x-axis and the y-axis
- `...`: further arguments passed to or from other methods

**Value**

Returns a list of class ’coinertia’, sub-class ’dudi’ containing:

- `call`
- `rank`
- `nf`: a numeric value indicating the number of kept axes
- `RV`: a numeric value, the RV coefficient
eig
a numeric vector with all the eigenvalues

lw
a numeric vector with the rows weigths (crossed table)

cw
a numeric vector with the columns weigths (crossed table)

tab
a crossed table (CT)

li
CT row scores (cols of dudiY)

li
Principal components (loadings for cols of dudiY)

c0
CT col scores (cols of dudiX)

c1
Principal axes (cols of dudiX)

lX
Row scores (rows of dudiX)

mX
Normed row scores (rows of dudiX)

lY
Row scores (rows of dudiY)

mY
Normed row scores (rows of dudiY)

aX
Correlations between dudiX axes and coinertia axes

aY
Correlations between dudiY axes and coinertia axes

WARNING
IMPORTANT: dudi1 and dudi2 must have identical row weights.

Author(s)
Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples
\begin{verbatim}
data(doubs)
dudi1 <- dudi.pca(doubs$env, scale = TRUE, scan = FALSE, nf = 3)
dudi2 <- dudi.pca(doubs$fish, scale = FALSE, scan = FALSE, nf = 2)
coin1 <- coinertia(dudi1,dudi2, scan = FALSE, nf = 2)
coin1
summary(coin1)
\end{verbatim}

\begin{verbatim}
if(adegraphicsLoaded()) {
g1 <- s.arrow(coin1$l1, plab.cex = 0.7)
g2 <- s.arrow(coin1$c1, plab.cex = 0.7)
g3 <- s.corcircle(coin1$aX, plot = FALSE)
g4 <- s.corcircle(coin1$aY, plot = FALSE)
}
cbindADEg(g3, g4, plot = TRUE)
g5 <- plot(coin)

) else {
  s.arrow(coin$l1, clab = 0.7)
  s.arrow(coin$c1, clab = 0.7)
  par(mfrow = c(1,2))
  s.corcircle(coin$aX)
  s.corcircle(coin$aY)
  par(mfrow = c(1,1))
  plot(coin)
}

---

coleo Table of Fuzzy Biological Traits

Description

This data set coleo (coleoptera) is a a fuzzy biological traits table.

Usage

data(coleo)

Format

coleo is a list of 5 components.

  tab is a data frame with 110 rows (species) and 32 columns (categories).
  species.names is a vector of species names.
  moda.names is a vector of fuzzy variables names.
  families is a factor species family.
  col.blocks is a vector containing the number of categories of each trait.

Source


Examples

data(coleo)
op <- par(no.readonly = TRUE)
coleo.fuzzy <- prep.fuzzy.var(coleo$tab, coleo$col.blocks)
fca1 <- dudi.fca(coleo.fuzzy, sca = FALSE, nf = 3)
indica <- factor(rep(names(coleo$col), coleo$col))
if(adegraphicsLoaded()) {
  glist <- list()
  for(i in levels(indica)) {
    df <- coleo$tab[, which(indica == i)]
    names(df) <- coleo$moda.names[which(indica == i)]
    glist[i] <- s.distri(fca1$l1, df, psub.text = as.character(i), ellipseSize = 0,
                         starSize = 0.5, plot = FALSE, storeData = TRUE)
  }
  G <- ADEgS(glist, layout = c(3, 3))
} else {
  par(mfrow = c(3, 3))
  for(j in levels(indica))
    s.distri(fca1$l1, coleo$tab[, which(indica == j)], clab = 1.5, sub = as.character(j),
             cell = 0, csta = 0.5, csub = 3, label = coleo$moda.names[which(indica == j)])
  par(op)
  par(mfrow = c(1, 1))
}

combine.4thcorner Functions to combine and adjust the outputs 3-table methods

Description

Functions to combine and adjust the outputs of the fourthcorner and randtest.rlq functions created using permutational models 2 and 4 (sequential approach).

Usage

combine.randtest.rlq(obj1, obj2, ...)
combine.4thcorner(four1,four2)
p.adjust.4thcorner(x, p.adjust.method.G = p.adjust.methods,
                   "levels"))

Arguments

four1 an object of the class 4thcorner created with modeltype = 2 (or 4)
four2 an object of the class 4thcorner created with modeltype = 4 (or 2)
obj1 an object created with randtest.rlq and modeltype = 2 (or 4)
obj2 an object created with randtest.rlq and modeltype = 4 (or 2)
x an object of the class 4thcorner
p.adjust.method.G
   a string indicating a method for multiple adjustment used for output tabG, see
   p.adjust.methods for possible choices
p.adjust.method.D
   a string indicating a method for multiple adjustment used for output tabD/tabD2,
   see p.adjust.methods for possible choices
combine.4thcorner

p.adjust.D a string indicating if multiple adjustment for tabD/tabD2 should be done globally or only between levels of a factor ("levels", as in the original paper of Legendre et al. 1997)

... further arguments passed to or from other methods

Details

The functions combines the outputs of two objects (created by fourthcorner and randtest.rlq functions) as described in Dray and Legendre (2008) and ter Braak et al (2012).

Value

The functions return objects of the same class than their argument. They simply create a new object where pvalues are equal to the maximum of pvalues of the two arguments.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also

rlq, fourthcorner, p.adjust.methods

Examples

data(aravo)
four2 <- fourthcorner(aravo$env, aravo$spe, aravo$traits, nrepet=99,modeltype=2)
four4 <- fourthcorner(aravo$env, aravo$spe, aravo$traits, nrepet=99,modeltype=4)
four.comb <- combine.4thcorner(four2,four4)
## or directly:
## four.comb <- fourthcorner(aravo$env, aravo$spe, aravo$traits, nrepet=99,modeltype=6)
summary(four.comb)
plot(four.comb, stat = "G")
corkdist

Tests of randomization between distances applied to 'kdist' objects

Description

The mantelkdist and RVkdist functions apply to blocks of distance matrices the mantel.rtest and RV.rtest functions.

Usage

mantelkdist (kd, nrepet = 999, ...)
RVkdist (kd, nrepet = 999, ...)
## S3 method for class 'corkdist'
plot(x, whichinrow = NULL, whichincol = NULL,
gap = 4, nclass = 10, ...)

Arguments

kd  
a list of class kdist

nrepet  
the number of permutations

x  
an objet of class corkdist, coming from RVkdist or mantelkdist

whichinrow  
a vector of integers to select the graphs in rows (if NULL all the graphs are computed)

whichincol  
a vector of integers to select the graphs in columns (if NULL all the graphs are computed)

gap  
an integer to determinate the space between two graphs

nclass  
a number of intervals for the histogram

...  
further arguments passed to or from other methods

Details

The corkdist class has some generic functions print, plot and summary. The plot shows bivariate scatterplots between semi-matrices of distances or histograms of simulated values with an error position.

Value

a list of class corkdist containing for each pair of distances an object of class randtest (permutation tests).

Author(s)

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
Examples

data(friday87)
fri.w <- ktab.data.frame(friday87$fau, friday87$fau.blo, tabnames = friday87$tab.names)
fri.kc <- lapply(1:10, function(x) dist.binary(fri.w[[x]], 10))
names(fri.kc) <- substr(friday87$tab.names, 1, 4)
fri.kd <- kdist(fri.kc)
fri.mantel <- mantelkdist(kd = fri.kd, nrepet = 999)

plot(fri.mantel, 1:5, 1:5)
plot(fri.mantel, 1:5, 6:10)
plot(fri.mantel, 6:10, 1:5)
plot(fri.mantel, 6:10, 6:10)
s.corcircle(dudi.pca(as.data.frame(fri.kd), scan = FALSE)$co)
plot(RVkdist(fri.kd), 1:5, 1:5)

data(yanomama)
m1 <- mantelkdist(kdist(yanomama), 999)
m1
summary(m1)
plot(m1)

---

**corvus**  
*Corvus morphology*

**Description**

This data set gives a morphological description of 28 species of the genus Corvus split in two habitat types and phylogeographic stocks.

**Usage**

data(corvus)

**Format**

corvus is data frame with 28 observations (the species) and 4 variables:

- **wing** : wing length (mm)
- **bill** : bill length (mm)
- **habitat** : habitat with two levels clos and open
- **phylog** : phylogeographic stock with three levels amer(America), orien(Oriental-Australian), pale(Paleoarctic-African)

**References**

Examples

data(corvus)

if(adegraphicsLoaded()) {
  g1 <- s.label(corvus[, 1:2], plab.cex = 0, porigin.include = FALSE, pgrid.draw = FALSE,
               paxes.draw = TRUE, paxes.asp = "full", xlab = names(corvus)[2],
               ylab = names(corvus)[2], plot = FALSE)
  g2 <- s.class(corvus[, 1:2], corvus[, 4]:corvus[, 3], plot = FALSE)
  G <- superpose(g1, g2, plot = TRUE)
} else {
  plot(corvus[, 1:2])
  s.class(corvus[, 1:2], corvus[, 4]:corvus[, 3], add.p = TRUE)
}

---

costatis

STATIS and Co-Inertia: Analysis of a series of paired ecological tables

Description

Analysis of a series of pairs of ecological tables. This function uses Partial Triadic Analysis (pta) and coinertia to do the computations.

Usage

costatis(KTX, KTY, scannf = TRUE)

Arguments

KTX

an objet of class ktab

KTY

an objet of class ktab

scannf

a logical value indicating whether the eigenvalues bar plot should be displayed

Details

This function takes 2 ktabs. It does a PTA (partial triadic analysis: pta) on each ktab, and does a coinertia analysis (coinertia) on the compromises of the two PTAs.

Value

a list of class coinertia, subclass dudi. See coinertia

WARNING

IMPORTANT: KTX and KTY must have the same k-tables structure, the same number of columns, and the same column weights.
costatis.randtest

Author(s)
Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

References

Examples
```r
data(meau)
wit1 <- withinpca(meau$env, meau$design$season, scan = FALSE, scal = "total")
pcaspe <- dudi.pca(meau$spe, scale = FALSE, scan = FALSE, nf = 2)
wit2 <- wca(pcaspe, meau$design$season, scan = FALSE, nf = 2)
kta1 <- ktab.within(wit1, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kta2 <- ktab.within(wit2, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
costatis1 <- costatis(kta1, kta2, scan = FALSE)
plot(costatis1)
```

costatis.randtest  Monte-Carlo test on a Costatis analysis (in C).

Description
Performs a Monte-Carlo test on a Costatis analysis.

Usage
costatis.randtest(KTX, KTY, nrepet = 999, ...)

Arguments

- **KTX**  
an objet of class ktab
- **KTY**  
an objet of class ktab
- **nrepet**  
the number of permutations
- **...**  
further arguments passed to or from other methods

Value

a list of the class randtest

Author(s)
Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>
References


Examples

data(meau)
wit1 <- withinpca(meau$env, meau$design$season, scan = FALSE, scal = "total")
pcaspe <- dudi.pca(meau$spe, scale = FALSE, scan = FALSE, nf = 2)
wit2 <- wca(pcaspe, meau$design$season, scan = FALSE, nf = 2)
kta1 <- ktab.within(wit1, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kta2 <- ktab.within(wit2, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
costatis1 <- costatis(kta1, kta2, scan = FALSE)
costatis.randtest(kta1, kta2)

---

dagnelie.test  Dagnelie multinormality test

Description

Compute Dagnelie test of multivariate normality on a data table of n objects (rows) and p variables (columns), with n > (p+1).

Usage

dagnelie.test(x)

Arguments

x  Multivariate data table (matrix or data.frame).

Details

Dagnelie’s goodness-of-fit test of multivariate normality is applicable to multivariate data. Mahalanobis generalized distances are computed between each object and the multivariate centroid of all objects. Dagnelie’s approach is that, for multinormal data, the generalized distances should be normally distributed. The function computes a Shapiro-Wilk test of normality of the Mahalanobis distances; this is our improvement of Dagnelie’s method. The null hypothesis (H0) is that the data are multinormal, a situation where the Mahalanobis distances should be normally distributed. In that case, the test should not reject H0, subject to type I error at the selected significance level.

Numerical simulations by D. Borcard have shown that the test had correct levels of type I error for values of n between 3p and 8p, where n is the number of objects and p is the number of variables in the data matrix (simulations with 1 <= p <= 100). Outside that range of n values, the results were too liberal, meaning that the test rejected too often the null hypothesis of normality. For p = 2, the simulations showed the test to be valid for 6 <= n <= 13 and too liberal outside that range. If H0 is not rejected in a situation where the test is too liberal, the result is trustworthy.
Calculation of the Mahalanobis distances requires that \( n > p+1 \) (actually, \( n > \text{rank}+1 \)). With fewer objects (\( n \)), all points are at equal Mahalanobis distances from the centroid in the resulting space, which has \( \min(\text{rank}, (n-1)) \) dimensions. For data matrices that happen to be collinear, the function uses \( \text{ginv} \) for inversion.

This test is not meant to be used with univariate data; in simulations, the type I error rate was higher than the 5% significance level for all values of \( n \). Function \text{shapiro.test} should be used in that situation.

**Value**

A list containing the following results:

- **Shapiro.Wilk**: \( W \) statistic and p-value
- **dim**: dimensions of the data matrix, \( n \) and \( p \)
- **rank**: the rank of the covariance matrix
- **D**: Vector containing the Mahalanobis distances of the objects to the multivariate centroid

**Author(s)**

Daniel Borcard and Pierre Legendre

**References**


**Examples**

```r
# Example 1: 2 variables, n = 100
n <- 100; p <- 2
mat <- matrix(rnorm(n*p), n, p)
(out <- dagnelie.test(mat))

# Example 2: 10 variables, n = 50
n <- 50; p <- 10
mat <- matrix(rnorm(n*p), n, p)
(out <- dagnelie.test(mat))

# Example 3: 10 variables, n = 100
n <- 100; p <- 10
mat <- matrix(rnorm(n*p), n, p)
(out <- dagnelie.test(mat))

# Plot a histogram of the Mahalanobis distances
```

hist(out$D)

# Example 4: 10 lognormal random variables, n = 50
n <- 50; p <- 10
mat <- matrix(round(exp(rnorm((n*p), mean = 0, sd = 2.5))), n, p)
(out <- dagnelie.test(mat))
# Plot a histogram of the Mahalanobis distances
hist(out$D)

---

### Deprecated functions

**Deprecated functions in ade4**

**Description**

The functions/data listed below are deprecated. The R code of the deprecated functions are stored for memory in the file ade4-deprecated.R.

- between: replaced by bca
- betweencoinertia: replaced by bca.coinertia
- char2genet: replaced by df2genind and genind2genpop in the adegenet package
- count2genet: replaced by df2genind and genind2genpop in the adegenet package
- dist.genet: replaced by dist.genpop in the adegenet package
- EH: replaced by EH in the adiv package
- freq2genet: replaced by df2genind and genind2genpop in the adegenet package
- fuzzygenet: replaced by df2genind in the adegenet package
- optimEH: replaced by optimEH in the adiv package
- orisaved: replaced by orisaved in the adiv package
- orthogram: replaced by orthogram in the adephylo package
- randEH: replaced by randEH in the adiv package
- within: replaced by wca
- withincoinertia: replaced by wca.coinertia

---

### deug

**Exam marks for some students**

**Description**

This data set gives the exam results of 104 students in the second year of a French University onto 9 subjects.

**Usage**

data(deug)
Format

deg is a data frame with 104 students and 9 subjects: Algebra, Analysis, Proba, Informatic, Economy, Option1, Option2, English, Sport.

**tab** is a data frame with 104 students and 9 subjects: Algebra, Analysis, Proba, Informatic, Economy, Option1, Option2, English, Sport.

**result** is a factor of 104 components giving the final exam levels (A+, A, B, B-, C-, D).

**cent** is a vector of required marks by subject to get exactly 10/20 with a coefficient.

Source

University of Lyon 1

Examples

data(deug)

# decentred PCA
pca1 <- dudi.pca(deug$tab, scal = FALSE, center = deug$cent, scan = FALSE)

if(adegraphicsLoaded()) {
  g1 <- s.class(pca1$li, deug$result, plot = FALSE)
  g2 <- s.arrow(40 * pca1$c1, plot = FALSE)
  G <- superpose(g1, g2, plot = TRUE)
} else {
  s.class(pca1$li, deug$result)
  s.arrow(40 * pca1$c1, add.plot = TRUE)
}

---

disc [Rao’s dissimilarity coefficient]

disc

Description

Calculates the root square of Rao’s dissimilarity coefficient between samples.

Usage

disc(samples, dis = NULL, structures = NULL)

Arguments

samples a data frame with elements as rows, samples as columns, and abundance, presence-absence or frequencies as entries.

dis an object of class dist containing distances or dissimilarities among elements. If dis is NULL, equidistances are used.

structures a data frame containing, in the jth row and the kth column, the name of the group of level k to which the jth population belongs.
**discrimin**

Linear Discriminant Analysis (descriptive statistic)

**Description**

performs a linear discriminant analysis.

**Value**

Returns a list of objects of class dist

**Author(s)**

Sandrine Pavoine <pavoine@mnhn.fr>

**References**


**Examples**

```r
data(humDNAm)
humDNA.dist <- disc(humDNAm$samples, sqrt(humDNAm$distances), humDNAm$structures)
humDNA.dist
is.euclid(humDNA.dist$samples)
is.euclid(humDNA.dist$regions)

## Not run:
data(ecomor)
dtaxo <- dist.taxo(ecomor$taxo)
ecomor.dist <- disc(ecomor$habitat, dtaxo)
ecomor.dist
is.euclid(ecomor.dist)

## End(Not run)
```

**Usage**

```r
discrimin(dudi, fac, scannf = TRUE, nf = 2)
## S3 method for class 'discrimin'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'discrimin'
print(x, ...)
```
**Arguments**

- `dudi` a duality diagram, object of class `dudi`
- `fac` a factor defining the classes of discriminant analysis
- `scannf` a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf` if `scannf` FALSE, an integer indicating the number of kept axes
- `x` an object of class ‘discrimin’
- `xax` the column number of the x-axis
- `yax` the column number of the y-axis
- `...` further arguments passed to or from other methods

**Value**

returns a list of class ‘discrimin’ containing:

- `nf` a numeric value indicating the number of kept axes
- `eig` a numeric vector with all the eigenvalues
- `fa` a matrix with the loadings: the canonical weights
- `li` a data frame which gives the canonical scores
- `va` a matrix which gives the cosines between the variables and the canonical scores
- `cp` a matrix which gives the cosines between the components and the canonical scores
- `gc` a data frame which gives the class scores

**Author(s)**

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

**See Also**

`lda` in package MASS

**Examples**

data(chazeb)
dis1 <- discrimin(dudi.pca(chazeb$tab, scan = FALSE), chazeb$cla,
  scan = FALSE)
dis1
if(!adegraphicsLoaded())
  plot(dis1)

data(skulls)
plot(discrimin(dudi.pca(skulls, scan = FALSE), gl(5,30),
  scan = FALSE))
discrimin.coa  Discriminant Correspondence Analysis

Description

performs a discriminant correspondence analysis.

Usage

discrimin.coa(df, fac, scannf = TRUE, nf = 2)

Arguments

df               a data frame containing positive or null values
fac              a factor defining the classes of discriminant analysis
scannf           a logical value indicating whether the eigenvalues bar plot should be displayed
nf               if scannf FALSE, an integer indicating the number of kept axes

Value

a list of class discrimin. See discrimin

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(perthi02)
plot(discrimin.coa(perthi02$tab, perthi02$cla, scan = FALSE))
Computation of Distance Matrices for Binary Data

Description

computes for binary data some distance matrices.

Usage

\[
\text{dist.binary}(\text{df}, \text{method} = \text{NULL}, \text{diag} = \text{FALSE}, \text{upper} = \text{FALSE})
\]

Arguments

- **df**: a matrix or a data frame with positive or null numeric values. Used with \(\text{as.matrix}(1 \times (\text{df} > 0))\)
- **method**: an integer between 1 and 10. If NULL the choice is made with a console message. See details
- **diag**: a logical value indicating whether the diagonal of the distance matrix should be printed by `print.dist`
- **upper**: a logical value indicating whether the upper triangle of the distance matrix should be printed by `print.dist`

Details

Let be the contingency table of binary data such as \(n_{11} = a, n_{10} = b, n_{01} = c\) and \(n_{00} = d\). All these distances are of type \(d = \sqrt{1 - s}\) with \(s\) a similarity coefficient.

1. **Jaccard index (1901)**  
   \(S_1 = \frac{a}{a + b + c}\)

2. **Simple matching coefficient of Sokal & Michener (1958)**  
   \(S_2 = \frac{a + d}{a + b + c + d}\)

3. **Sokal & Sneath (1963)**  
   \(S_3 = \frac{a}{a + 2(b + c)}\)

4. **Rogers & Tanimoto (1960)**  
   \(S_4 = \frac{a + d}{(a + 2(b + c) + d)}\)

5. **Dice (1945) or Sorensen (1948)**  
   \(S_5 = \frac{2a}{2a + b + c}\)

6. **Hamann coefficient**  
   \(S_6 = \frac{2(b + c) + d}{a + b + c + d}\)

7. **Ochiai (1957)**  
   \(S_7 = \frac{a}{\sqrt{(a+b)(a+c)}}\)

8. **Sokal & Sneath (1963)**  
   \(S_8 = \frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}\)

9. **Phi of Pearson**  
   \(S_9 = \frac{ad - bc}{\sqrt{(a+b)(a+c)(b+d)(d+c)}}\)

10. **S2 coefficient of Gower & Legendre**  
    \(S_{10} = \frac{a}{a + b + c + d}\)

Value

returns a distance matrix of class `dist` between the rows of the data frame
Author(s)
Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References

Examples
data(aviurba)
for (i in 1:10) {
  d <- dist.binary(aviurba$fau, method = i)
  cat(attr(d, "method"), is.euclid(d), "n")
}

---

**dist.dudi**

*Computation of the Distance Matrix from a Statistical Triplet*

**Description**
computes for a statistical triplet a distance matrix.

**Usage**
dist.dudi(dudi, amongrow = TRUE)

**Arguments**
- **dudi** a duality diagram, object of class dudi
- **amongrow** a logical value computing the distance if TRUE, between rows, if FALSE between columns.

**Value**
an object of class dist

**Author(s)**
Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>

**Examples**
data (meaudret)
 pca1 <- dudi.pca(meaudret$env, scan = FALSE)
 sum((dist(scalewt(meaudret$env)) - dist.dudi(pca1))^2)
#[1] 4.045e-29 the same thing
dist.ktab  Mixed-variables coefficient of distance

Description

The mixed-variables coefficient of distance generalizes Gower's general coefficient of distance to allow the treatment of various statistical types of variables when calculating distances. This is especially important when measuring functional diversity. Indeed, most of the indices that measure functional diversity depend on variables (traits) that have various statistical types (e.g. circular, fuzzy, ordinal) and that go through a matrix of distances among species.

Usage

dist.ktab(x, type, option = c("scaledBYrange", "scaledBYsd", "noscale"),
scann = FALSE, tol = 1e-8)
ldist.ktab(x, type, option = c("scaledBYrange", "scaledBYsd",
"noscale"), scann = FALSE, tol = 1e-8)
kdist.cor(x, type, option = c("scaledBYrange", "scaledBYsd", "noscale"),
scann = FALSE, tol = 1e-8, squared = TRUE)
prep.fuzzy(df, col.blocks, row.w = rep(1, nrow(df)), labels = paste("F",
1:length(col.blocks), sep = ""))
prep.binary(df, col.blocks, labels = paste("B", 1:length(col.blocks), sep = ""))
prep.circular(df, rangemin = apply(df, 2, min, na.rm = TRUE), rangemax =
apply(df, 2, max, na.rm = TRUE))

Arguments

x Object of class ktab (see details)
type Vector that provide the type of each table in x. The possible types are "Q" (quantitative), "O" (ordinal), "N" (nominal), "D" (dichotomous), "F" (fuzzy, or expressed as a proportion), "B" (multichoice nominal variables, coded by binary columns), "C" (circular). Values in type must be in the same order as in x.
option A string that can have three values: either "scaledBYrange" if the quantitative variables must be scaled by their range, or "scaledBYsd" if they must be scaled by their standard deviation, or "noscale" if they should not be scaled. This last option can be useful if the the values have already been normalized by the known range of the whole population instead of the observed range measured on the sample. If x contains data from various types, then the option "scaledBYsd" is not suitable (a warning will appear if the option selected with that condition).
scann A logical. If TRUE, then the user will have to choose among several possible functions of distances for the quantitative, ordinal, fuzzy and binary variables.
tol A tolerance threshold: a value less than tol is considered as null.
squared A logical, if TRUE, the squared distances are considered.
df Objet of class data.frame
col.blocks A vector that contains the number of levels per variable (in the same order as in df)
row.w A vector of row weights
labels the names of the traits
rangemin A numeric corresponding to the smallest level where the loop starts
rangemax A numeric corresponding to the highest level where the loop closes

Details

When preparing the object of class ktab (object x), variables of type "Q", "O", "D", "F", "B" and "C" should be of class numeric (the class ordered is not yet considered by dist.ktab); variables of type "N" should be of class character or factor

Value

The functions provide the following results:

dist.ktab returns an object of class dist;
ldist.ktab returns a list of objects of class dist that correspond to the distances between species calculated per trait;
kdist.cor returns a list of three objects: "paircov" provides the covariance between traits in terms of (squared) distances between species; "paircor" provides the correlations between traits in terms of (squared) distances between species; "glocor" provides the correlations between the (squared) distances obtained for each trait and the global (squared) distances obtained by mixing all the traits (= contributions of traits to the global distances);
prep.binary and prep.fuzzy returns a data frame with the following attributes: col.blocks specifies the number of columns per fuzzy variable; col.num specifies which variable each column belongs to;
prep.circular returns a data frame with the following attributes: max specifies the number of levels in each circular variable.

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>

References


See Also
daisy in the case of ratio-scale (quantitative) and nominal variables; and woangers for an application.

Examples

# With fuzzy variables
data(bsetal97)

w <- prep.fuzzy(bsetal97$biol, bsetal97$biol.blo)
w[1:6, 1:10]
ktab1 <- ktab.list.df(list(w))
dis <- dist.ktab(ktab1, type = "F")
as.matrix(dis)[1:5, 1:5]

## Not run:
# With ratio-scale and multichoice variables
data(ecomor)

wM <- log(ecomor$morpho + 1) # Quantitative variables
wD <- ecomor$diet
# wD is a data frame containing a multichoice nominal variable
# (diet habit), with 8 modalities (Granivorous, etc)
# We must prepare it by prep.binary
head(wD)
wD <- prep.binary(wD, col.blocks = 8, label = "diet")
wF <- ecomor$forsub
# wF is also a data frame containing a multichoice nominal variable
# (foraging substrat), with 6 modalities (Foliage, etc)
# We must prepare it by prep.binary
head(wF)
wF <- prep.binary(wF, col.blocks = 6, label = "foraging")
# Another possibility is to combine the two last data frames wD and wF as
# they contain the same type of variables
wB <- cbind.data.frame(ecomor$diet, ecomor$forsub)
head(wB)
wB <- prep.binary(wB, col.blocks = c(8, 6), label = c("diet", "foraging"))
# The results given by the two alternatives are identical
ktab2 <- ktab.list.df(list(wM, wD, wF))
disecomor <- dist.ktab(ktab2, type = c("Q", "B", "B"))
as.matrix(disecomor)[1:5, 1:5]
contrib2 <- kdist.cor(ktab2, type = c("Q", "B", "B"))

ktab3 <- ktab.list.df(list(wM, wB))
disecomor2 <- dist.ktab(ktab3, type = c("Q", "B"))
as.matrix(disecomor2)[1:5, 1:5]
contrib3 <- kdist.cor(ktab3, type = c("Q", "B"))

# With a range of variables
data(woangers)
traits <- woangers$traits
# Nominal variables 'll', 'pr', 'lp' and 'le'
# (see table 1 in the main text for the codes of the variables)
tabN <- traits[,c(1:2, 7, 8)]
# Circular variable 'fo'
tabC <- traits[3]
tabCp <- prep.circular(tabC, 1, 12)
# The levels of the variable lie between 1 (January) and 12 (December).
# Ordinal variables 'he', 'ae' and 'un'
tabO <- traits[, 4:6]
# Fuzzy variables 'mp', 'pe' and 'di'
tabF <- traits[, 9:19]
tabFp <- prep.fuzzy(tabF, c(3, 3, 5), labels = c("mp", "pe", "di"))
# 'mp' has 3 levels, 'pe' has 3 levels and 'di' has 5 levels.
# Quantitative variables 'lo' and 'lf'
tabQ <- traits[, 20:21]
ktab1 <- ktab.list.df(list(tabN, tabCp, tabO, tabFp, tabQ))
distrait <- dist.ktab(ktab1, c("N", "C", "O", "F", "Q"))
is.euclid(distrait)
contrib <- kdist.cor(ktab1, type = c("N", "C", "O", "F", "Q"))
contrib
dotchart(sort(contrib$glocor), labels = rownames(contrib$glocor)[order(contrib$glocor[, 1])])

## End(Not run)

---

dist.neig

Computation of the Distance Matrix associated to a Neighbouring Graph

Description

This distance matrix between two points is the length of the shortest path between these points.

Usage

dist.neig(neig)

Arguments

neig a neighbouring graph, object of class neig

Value

returns a distance matrix, object of class dist

Author(s)

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
dist.prop

Examples

```r
data(elec88)
d0 <- dist.neig(elec88$neig)
plot(dist(elec88$xy),d0)
```

Description

computes for percentage data some distance matrices.

Usage

```r
dist.prop(df, method = NULL, diag = FALSE, upper = FALSE)
```

Arguments

- `df`: a data frame containing only positive or null values, used as row percentages
- `method`: an integer between 1 and 5. If NULL the choice is made with a console message. See details
- `diag`: a logical value indicating whether the diagonal of the distance matrix should be printed by `print.dist`
- `upper`: a logical value indicating whether the upper triangle of the distance matrix should be printed by `print.dist`

Details

1 = Manly  
\[ d_1 = \frac{1}{2} \sum_{i=1}^{K} |p_i - q_i| \]

2 = Overlap index Manly  
\[ d_2 = 1 - \frac{\sum_{i=1}^{K} p_i q_i}{\sqrt{\sum_{i=1}^{K} p_i^2 \sqrt{\sum_{i=1}^{K} q_i^2}}} \]

3 = Rogers 1972 (one locus)  
\[ d_3 = \sqrt{\frac{1}{2} \sum_{i=1}^{K} (p_i - q_i)^2} \]

4 = Nei 1972 (one locus)  
\[ d_4 = \ln \left( \frac{\sum_{i=1}^{K} p_i q_i}{\sqrt{\sum_{i=1}^{K} p_i^2 \sqrt{\sum_{i=1}^{K} q_i^2}}} \right) \]

5 = Edwards 1971 (one locus)  
\[ d_5 = \sqrt{1 - \sum_{i=1}^{K} \sqrt{p_i q_i}} \]

Value

returns a distance matrix, object of class `dist`

Author(s)

Daniel Chessel  
Stéphane Dray <stephane.dray@univ-lyon1.fr>
References


Examples

data(microsatt)
w <- microsatt$tab[1:microsatt$loci.eff[1]]

if(adegraphicsLoaded()) {
  g1 <- scatter(dudi.pco(lingoes(dist.prop(w, 1)), scann = FALSE), plot = FALSE)
  g2 <- scatter(dudi.pco(lingoes(dist.prop(w, 2)), scann = FALSE), plot = FALSE)
  g3 <- scatter(dudi.pco(dist.prop(w, 3), scann = FALSE), plot = FALSE)
  g4 <- scatter(dudi.pco(lingoes(dist.prop(w, 4)), scann = FALSE), plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  scatter(dudi.pco(lingoes(dist.prop(w, 1)), scann = FALSE))
  scatter(dudi.pco(lingoes(dist.prop(w, 2)), scann = FALSE))
  scatter(dudi.pco(dist.prop(w, 3), scann = FALSE))
  scatter(dudi.pco(lingoes(dist.prop(w, 4)), scann = FALSE))
  par(mfrow = c(1, 1))
}

dist.quant

Computation of Distance Matrices on Quantitative Variables

Description

computes on quantitative variables, some distance matrices as canonical, Joreskog and Mahalanobis.

Usage

dist.quant(df, method = NULL, diag = FALSE, upper = FALSE, tol = 1e-07)

Arguments

df a data frame containing only quantitative variables
method an integer between 1 and 3. If NULL the choice is made with a console message. See details
dist.quant

- **diag**: a logical value indicating whether the diagonal of the distance matrix should be printed by 'print.dist'
- **upper**: a logical value indicating whether the upper triangle of the distance matrix should be printed by 'print.dist'
- **tol**: used in case 3 of method as a tolerance threshold for null eigenvalues

**Details**

All the distances are of type

\[ d = \| x - y \|_A = \sqrt{(x - y)^t A (x - y)} \]

- **1 = Canonical**: \( A = \text{Identity} \)
- **2 = Joreskog**: \( A = \frac{1}{\text{diag(cov)}} \)
- **3 = Mahalanobis**: \( A = \text{inv(cov)} \)

**Value**

an object of class `dist`

**Author(s)**

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>

**Examples**

```r
data(ecomor)

if(adeGraphicsLoaded()) {
  g1 <- scatter(dudi.pco(dist.quant(ecomor$morpho, 3), scan = FALSE), plot = FALSE)
  g2 <- scatter(dudi.pco(dist.quant(ecomor$morpho, 2), scan = FALSE), plot = FALSE)
  g3 <- scatter(dudi.pco(dist(scalewt(ecomor$morpho)), scan = FALSE), plot = FALSE)
  g4 <- scatter(dudi.pco(dist.quant(ecomor$morpho, 1), scan = FALSE), plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  scatter(dudi.pco(dist.quant(ecomor$morpho, 3), scan = FALSE))
  scatter(dudi.pco(dist.quant(ecomor$morpho, 2), scan = FALSE))
  scatter(dudi.pco(dist(scalewt(ecomor$morpho)), scan = FALSE))
  scatter(dudi.pco(dist.quant(ecomor$morpho, 1), scan = FALSE))
  par(mfrow = c(1, 1))
}
```
divc

Rao’s diversity coefficient also called quadratic entropy

Description

Calculates Rao’s diversity coefficient within samples.

Usage

divc(df, dis, scale)

Arguments

- **df**: a data frame with elements as rows, samples as columns, and abundance, presence-absence or frequencies as entries
- **dis**: an object of class `dist` containing distances or dissimilarities among elements. If `dis` is NULL, Gini-Simpson index is performed.
- **scale**: a logical value indicating whether or not the diversity coefficient should be scaled by its maximal value over all frequency distributions.

Value

Returns a data frame with samples as rows and the diversity coefficient within samples as columns.

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>

References


Examples

data(ecomor)
dtaxo <- dist.taxo(ecomor$taxo)
divc(ecomor$habitat, dtaxo)

data(humDNAm)
divc(humDNAm$samples, sqrt(humDNAm$distances))
### divcmax

**Maximal value of Rao’s diversity coefficient also called quadratic entropy**

**Description**

For a given dissimilarity matrix, this function calculates the maximal value of Rao’s diversity coefficient over all frequency distribution. It uses an optimization technique based on Rosen’s projection gradient algorithm and is verified using the Kuhn-Tucker conditions.

**Usage**

```r
divcmax(dis, epsilon, comment)
```

**Arguments**

- **dis**: an object of class `dist` containing distances or dissimilarities among elements.
- **epsilon**: a tolerance threshold: a frequency is non null if it is higher than epsilon.
- **comment**: a logical value indicating whether or not comments on the optimization technique should be printed.

**Value**

Returns a list

- **value**: the maximal value of Rao’s diversity coefficient.
- **vectors**: a data frame containing four frequency distributions: `sim` is a simple distribution which is equal to $D_1$, `pro` is equal to $z^2$, where $z$ is the nonnegative eigenvector of the matrix containing the squared dissimilarities among the elements, `met` is equal to $z^2$, `num` is a frequency vector maximizing Rao’s diversity coefficient.

**Author(s)**

Stéphane Champely <Stephane.Champely@univ-lyon1.fr>
Sandrine Pavoine <pavoine@mnhn.fr>

**References**


Examples

data(elec88)

# Dissimilarity matrix.
d0 <- dist(elec88$xy/100)

# Frequency distribution maximizing spatial diversity in France
# according to Rao's quadratic entropy.
France.m <- divcmax(d0)
w0 <- France.m$vectors$num
v0 <- France.m$value
idx <- (1:94)[w0 > 0]

if(!adegraphicsLoaded()) {
  # Smallest circle including all the 94 departments.
  # The squared radius of that circle is the maximal value of the
  # spatial diversity.
  w1 <- elec88$xy[idx, ]/100
  w.c <- apply(w1 * w0[idx], 2, sum)
  plot(elec88$xy[, 1]/100, elec88$xy[, 2]/100, asp=1)
  symbols(w.c[1], w.c[2], circles = sqrt(v0), inches = FALSE, add = TRUE)
  s.value(elec88$xy/100, w0, add.plot = TRUE)
}

dotchart.phylog

Representation of many quantitative variables in front of a phyloge-
netic tree

Description

dotchart.phylog represents the phylogenetic tree and draws Cleveland dot plot of each variable.

Usage

dotchart.phylog(phylog, values, y = NULL, scaling = TRUE, ranging =
TRUE, yranging = NULL, joining = TRUE, yjoining = NULL, ceti = 1, cdot =
1, csub = 1, f.phylog = 1/(1 + ncol(values)), ...)

Arguments

phylog an object of class phylog
values a vector or a data frame giving the variables
y a vector which values correspond to leaves positions
scaling if TRUE, data are scaled
ranging if TRUE, dotplots are drawn with the same horizontal limits
yranging a vector with two values giving the horizontal limits. If NULL, horizontal limits are defined by lower and upper values of data
joining 
if TRUE, segments join each point to a central value

yjoining 
a vector with the central value. If NULL, the central value equals 0

ct 
a character size for editing horizontal limits, used with `par("cex")*ct`

cdot 
a character size for plotting the points of the dot plot, used with `par("cex")*cdot`

csub 
a character size for editing the names of variables, used with `par("cex")*csub`

f.phylog 
a size coefficient for tree size (a parameter to draw the tree in proportion to leaves labels)

... 
further arguments passed to or from other methods

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

See Also

`symbols.phylog` and `table.phylog`

Examples

# one variable
tre <- c("((A,B),(C,D));")
phy <- newick2phylog(tre)
x <- 1:4
par(mfrow = c(2,2))
dotchart.phylog(phy, x, scaling = FALSE)
dotchart.phylog(phy, x)
dotchart.phylog(phy, x, joining = FALSE)
dotchart.phylog(phy, x, scaling = FALSE,
yjoining = 0, yranging = c(-1, 5))
par(mfrow = c(1,1))

# many variables
data(mjrochet)
phy <- newick2phylog(mjrochet$tre)
tab <- data.frame(log(mjrochet$tab))
dotchart.phylog(phy, tab, ceti = 0.5, csub = 0.6,
cleaves = 0, cdot = 0.6)
par(mfrow=c(1,1))
Description

This function represents \( n \) values on a circle. The \( n \) points are shared out regularly over the circle and put on the radius according to the value attributed to that measure.

Usage

\[
dotcircle(z, \text{alpha0} = \pi/2, \text{xlim} = \text{range(pretty(z))}, \\
\text{labels} = \text{names(z)}, \text{clabel} = 1, \text{cleg} = 1)
\]

Arguments

- \( z \): a numeric vector
- \( \text{alpha0} \): polar angle to put the first value
- \( \text{xlim} \): the ranges to be encompassed by the circle radius
- \( \text{labels} \): a vector of strings of characters for the angle labels
- \( \text{clabel} \): a character size for the labels, used with \( \text{par("cex")}*\text{clabel} \)
- \( \text{cleg} \): a character size for the ranges, used with \( \text{par("cex")}*\text{cleg} \)

Author(s)

Daniel Chessel

See Also

- \texttt{circ.plot}

Examples

\[
\begin{align*}
\text{w <- scores.neig(neig(n.cir = 24))} \\
\text{par(mfrow = c(4,4))} \\
\text{for (k in 1:16) dotcircle(w[,k], labels = 1:24)} \\
\text{par(mfrow = c(1,1))}
\end{align*}
\]
**doubs**

---

**Pair of Ecological Tables**

**Description**

This data set gives environmental variables, fish species and spatial coordinates for 30 sites.

**Usage**

```r
data(doubs)
```

**Format**

- `doubs` is a list with 4 components.
  - `env` is a data frame with 30 rows (sites) and 11 environmental variables.
  - `fish` is a data frame with 30 rows (sites) and 27 fish species.
  - `xy` is a data frame with 30 rows (sites) and 2 spatial coordinates.
  - `species` is a data frame with 27 rows (species) and 4 columns (names).

**Details**

The rows of `doubs$env`, `doubs$fish` and `doubs$xy` are 30 sites along the Doubs, a French and Switzerland river.

- `doubs$env` contains the following variables: dfs - distance from the source (km * 10), alt - altitude (m), slo \( \ln(x + 1) \) where \( x \) is the slope (per mil * 100), flo - minimum average stream flow (m\(^3\)s \(^{-1}\) * 100), pH (* 10), har - total hardness of water (mg/l of Calcium), pho - phosphates (mg/l * 100), nit - nitrates (mg/l * 100), amm - ammonia nitrogen (mg/l * 100), oxy - dissolved oxygen (mg/l * 10), bdo - biological demand for oxygen (mg/l * 10).


- `doubs$species` contains the names of the 27 fish species. The four columns correspond to: 1 = scientific name (Genus species), 2 = French common name, 3 = English common name, 4 = Four character code.

**Source**

References


Examples

```r
data(doubs)
pca1 <- dudi.pca(doubs$env, scan = FALSE)
pca2 <- dudi.pca(doubs$fish, scale = FALSE, scan = FALSE)
coiner1 <- coiner(pca1, pca2, scan = FALSE)

if(adegraphicsLoaded()) {
  g1 <- s.corcircle(coiner1$aX, plot = FALSE)
g2 <- s.value(doubs$xy, coiner1$lX[, 1], plot = FALSE)
g3 <- s.value(doubs$xy, coiner1$lX[, 2], plot = FALSE)
g4 <- s.arrow(coiner1$c1, plot = FALSE)
g5 <- s.match(coiner1$mX, coiner1$mY, plot = FALSE)
g6 <- s.corcircle(coiner1$aY, plot = FALSE)
g7 <- s.arrow(coiner1$l1, plot = FALSE)
g8 <- s.value(doubs$xy, coiner1$lY[, 1], plot = FALSE)
g9 <- s.value(doubs$xy, coiner1$lY[, 2], plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4, g5, g6, g7, g8, g9), layout = c(3, 3))
} else {
  par(mfrow = c(3, 3))
s.corcircle(coiner1$aX)
s.value(doubs$xy, coiner1$lX[, 1])
s.value(doubs$xy, coiner1$lX[, 2])
s.arrow(coiner1$c1)
s.match(coiner1$mX, coiner1$mY)
s.corcircle(coiner1$aY)
s.arrow(coiner1$l1)
s.value(doubs$xy, coiner1$lY[, 1])
s.value(doubs$xy, coiner1$lY[, 2])
par(mfrow = c(1, 1))
}
```

dpcoa

**Double principal coordinate analysis**

Description

Performs a double principal coordinate analysis

Usage

dpcoa(df, dis = NULL, scannf = TRUE, nf = 2, full = FALSE, tol = 1e-07, RaoDecomp = TRUE)
## S3 method for class 'dpcoa'
plot(x, xax = 1, yax = 2, ...)

## S3 method for class 'dpcoa'
print(x, ...)

## S3 method for class 'dpcoa'
summary(object, ...)

### Arguments

df
  a data frame with samples as rows and categories (i.e. species) as columns and abundance or presence-absence as entries. Previous releases of ade4 (<=1.6-2) considered the transposed matrix as argument.
dis
  an object of class dist containing the distances between the categories.
scanf
  a logical value indicating whether the eigenvalues bar plot should be displayed
RaoDecomp
  a logical value indicating whether Rao diversity decomposition should be performed
nf
  if scanf is FALSE, an integer indicating the number of kept axes
full
  a logical value indicating whether all non null eigenvalues should be kept
tol
  a tolerance threshold for null eigenvalues (a value less than tol times the first one is considered as null)
x, object
  an object of class dpcoa
xax
  the column number for the x-axis
yax
  the column number for the y-axis
...
  ... further arguments passed to or from other methods

### Value

Returns a list of class dpcoa containing:
call
  call
nf
  a numeric value indicating the number of kept axes
dw
  a numeric vector containing the weights of the elements (was w1 in previous releases of ade4)
1w
  a numeric vector containing the weights of the samples (was w2 in previous releases of ade4)
eig
  a numeric vector with all the eigenvalues
RaoDiv
  a numeric vector containing diversities within samples
RaoDis
  an object of class dist containing the dissimilarities between samples
RaoDecodiv
  a data frame with the decomposition of the diversity
dls
  a data frame with the coordinates of the elements (was l1 in previous releases of ade4)
li
  a data frame with the coordinates of the samples (was l2 in previous releases of ade4)
c1
  a data frame with the scores of the principal axes of the elements
Author(s)

Daniel Chessel
Sandrine Pavoine <pavoine@mnhn.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


Examples

data(humDNAm)
dpcoahum <- dpcoa(data.frame(t(humDNAm$samples)), sqrt(humDNAm$distances), scan = FALSE, nf = 2)
dpcoahum
if(adegraphicsLoaded()) {
  g1 <- plot(dpcoahum)
} else {
  plot(dpcoahum)
}

## Not run:
data(ecomor)
dtaxo <- dist.taxo(ecomor$taxo)
dpcoaeeco <- dpcoa(data.frame(t(ecomor$habitat)), dtaxo, scan = FALSE, nf = 2)
dpcoaeeco
if(adegraphicsLoaded()) {
  g1 <- plot(dpcoaeeco)
} else {
  plot(dpcoaeeco)
}

## End(Not run)

---

dudi

**Duality Diagram**

Description

as.dudi is called by many functions (dudi.pca, dudi.coa, dudi.acm, ...) and not directly by the user. It creates duality diagrams.
t.dudi returns an object of class 'dudi' where the rows are the columns and the columns are the rows of the initial dudi.
is.dudi returns TRUE if the object is of class dudi
redo.dudi computes again an analysis, eventually changing the number of kept axes. Used by other functions.
Usage

as.dudi(df, col.w, row.w, scannf, nf, call, type, tol = 1e-07,
full = FALSE)
## S3 method for class 'dudi'
print(x, ...) 
is.dudi(x)
redo.dudi(dudi, newnf = 2)
## S3 method for class 'dudi'
t(x)
## S3 method for class 'dudi'
summary(object, ...)
## S3 method for class 'dudi'
x[i,j]

Arguments

df a data frame with n rows and p columns
col.w a numeric vector containing the row weights
row.w a numeric vector containing the column weights
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes
call generally match.call()
type a string of characters: the returned list will be of class c(type, "dudi")
tol a tolerance threshold for null eigenvalues (a value less than tol times the first one is considered as null)
full a logical value indicating whether all non null eigenvalues should be kept
x, dudi, object objects of class dudi
... further arguments passed to or from other methods
newnf an integer indicating the number of kept axes
i,j elements to extract (integer or empty): index of rows (i) and columns (j)

Value

as.dudi and all the functions that use it return a list with the following components:

tab a data frame with n rows and p columns
cw column weights, a vector with n components
lw row (lines) weights, a vector with p components
eig eigenvalues, a vector with min(n,p) components
nf integer, number of kept axes
c1 principal axes, data frame with p rows and nf columns
l1 principal components, data frame with n rows and nf columns
c0 column coordinates, data frame with p rows and nf columns
li row coordinates, data frame with n rows and nf columns
call original call
Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


Examples

data(deug)
dd1 <- dudi.pca(deug$tab, scannf = FALSE)
dd1
t(dd1)
is.dudi(dd1)
redo.dudi(dd1,3)
summary(dd1)

---

Multiple Correspondence Analysis

dudi.acm

Description

dudi.acm performs the multiple correspondence analysis of a factor table.
acm.burt an utility giving the crossed Burt table of two factors table.
acm.disjonctif an utility giving the complete disjunctive table of a factor table.
boxplot.acm a graphic utility to interpret axes.

Usage

dudi.acm (df, row.w = rep(1, nrow(df)), scannf = TRUE, nf = 2)
acm.burt (df1, df2, counts = rep(1, nrow(df1)))
acm.disjonctif (df)
## S3 method for class 'acm'
boxplot(x, xax = 1, ...)

Arguments

df, df1, df2 data frames containing only factors
row.w, counts vector of row weights, by default, uniform weighting
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes
**dudi.acm**

x | an object of class acm
xax | the number of factor to display
... | further arguments passed to or from other methods

**Value**

dudi.acm returns a list of class acm and dudi (see dudi) containing

cr | a data frame which rows are the variables, columns are the kept scores and the values are the correlation ratios

**Author(s)**

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

**References**


**See Also**

s.chull, s.class

**Examples**

data(ours)
summary(ours)

if(adegraphicsLoaded()) {
  g1 <- sld.boxplot(dudi.acm(ours, scan = FALSE)$li[, 1], ours)
} else {
  boxplot(dudi.acm(ours, scan = FALSE))
}
## Not run:
data(banque)
banque.acm <- dudi.acm(banque, scann = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  g2 <- adegraphics:::scatter.dudi(banque.acm)
} else {
  scatter(banque.acm)
}

apply(banque.acm$cr, 2, mean)
banque.acm$eig[1:banque.acm$nf] # the same thing
if(adegraphicsLoaded()) {
  g3 <- s1d.boxplot(banque.acm$li[,1], banque)
  g4 <- scatter(banque.acm)
} else {
  boxplot(banque.acm)
  scatter(banque.acm)
}

s.value(banque.acm$li, banque.acm$li[,3])

bb <- acm.burt(banque, banque)
bbcoa <- dudi.coa(bb, scann = FALSE)
plot(banque.acm$c1[,1], bbcoa$c1[,1])  
# mca and coa of Burt table. Lebart & coll. section 1.4

bd <- acm.disjonctif(banque)
bdcoa <- dudi.coa(bd, scann = FALSE)
plot(banque.acm$li[,1], bdcoa$li[,1])
# mca and coa of disjonctive table. Lebart & coll. section 1.4
plot(banque.acm$co[,1], dudi.coa(bd, scann = FALSE)$co[,1])

## End(Not run)

---

**dudi.coa**

*Correspondence Analysis*

**Description**

performs a correspondence analysis.

**Usage**

dudi.coa(df, scannf = TRUE, nf = 2)

**Arguments**

- **df**
  a data frame containing positive or null values
- **scannf**
  a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**
  if scannf FALSE, an integer indicating the number of kept axes

**Value**

returns a list of class coa and dudi (see dudi) containing

- **N**
  the sum of all the values of the initial table
Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(rpjdl)
chisq.test(rpjdl$fau)$statistic
rpjdl.coa <- dudi.coa(rpjdl$fau, scannf = FALSE, nf = 4)
sum(rpjdl.coa$eig)*rpjdl.coa$N # the same

if(adegraphicsLoaded()) {
  g1 <- s.label(rpjdl.coa$co, plab.cex = 0.6, lab = rpjdl$frlab, plot = FALSE)
  g2 <- s.label(rpjdl.coa$li, plab.cex = 0.6, plot = FALSE)
  cbindADEg(g1, g2, plot = TRUE)
} else {
  par(mfrow = c(1,2))
  s.label(rpjdl.coa$co, clab = 0.6, lab = rpjdl$frlab)
  s.label(rpjdl.coa$li, clab = 0.6)
  par(mfrow = c(1,1))
}

data(bordeaux)
db <- dudi.coa(bordeaux, scan = FALSE)
db
score(db)

dudi.dec

Decentred Correspondence Analysis

Description

performs a decentred correspondence analysis.

Usage

dudi.dec(df, eff, scannf = TRUE, nf = 2)
Arguments

- **df**: a data frame containing positive or null values
- **eff**: a vector containing the reference distribution. Its length is equal to the number of rows of df
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes

Value

Returns a list of class `dec` and `dudi` (see `dudi`) containing also

- **R**: sum of all the values of the initial table

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

```r
data(ichtyo)
dudi1 <- dudi.dec(ichtyo$tab, ichtyo$eff, scan = FALSE)
sum(apply(ichtyo$tab, 2, function(x) chisq.test(x, p = ichtyo$eff/sum(ichtyo$eff))$statistic))
sum(dudi1$eig) * sum(ichtyo$eff) # the same
s.class(dudi1$li, ichtyo$dat, wt = ichtyo$eff/sum(ichtyo$eff))
```

Fuzzy Correspondence Analysis and Fuzzy Principal Components Analysis

Description

Theses functions analyse a table of fuzzy variables.

A fuzzy variable takes values of type $a = (a_1, \ldots, a_k)$ giving the importance of $k$ categories.

A missing data is denoted $(0, \ldots, 0)$.

Only the profile $a/\text{sum}(a)$ is used, and missing data are replaced by the mean profile of the others in the function `prep.fuzzy.var`. See ref. for details.
Usage

```r
prep.fuzzy.var (df, col.blocks, row.w = rep(1, nrow(df)))
dudi.fca(df, scannf = TRUE, nf = 2)
dudi.fpca(df, scannf = TRUE, nf = 2)
```

Arguments

- **df**: a data frame containing positive or null values
- **col.blocks**: a vector containing the number of categories for each fuzzy variable
- **row.w**: a vector of row weights
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes

Value

The function `prep.fuzzy.var` returns a data frame with the attribute `col.blocks`. The function `dudi.fca` returns a list of class `fca` and `dudi` (see `dudi`) containing also

- **cr**: a data frame which rows are the blocs, columns are the kept axes, and values are the correlation ratios.

The function `dudi.fpca` returns a list of class `pca` and `dudi` (see `dudi`) containing also

1. cent
2. norm
3. blo
4. indica
5. FST
6. inertia

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

```r
w1 <- matrix(c(1,0,0,2,1,0,2,0,2,2,2,0,0,1,1,1,1,3,1,0), 4, 5)
w1 <- data.frame(w1)
w2 <- prep.fuzzy.var(w1, c(2, 3))
w1
w2
attributes(w2)
```
data(bsetal97)
w <- prep.fuzzy.var(bsetal97$biol, bsetal97$biol.blo)

if(adegraphicsLoaded()) {
g1 <- plot(dudi.fca(w, scann = FALSE, nf = 3), plabels.cex = 1.5)
} else {
  scatter(dudi.fca(w, scann = FALSE, nf = 3), csub = 3, clab.moda = 1.5)
  scatter(dudi.fpca(w, scann = FALSE, nf = 3), csub = 3, clab.moda = 1.5)
}

## Not run:
w1 <- prep.fuzzy.var(bsetal97$biol, bsetal97$biol.blo)
w2 <- prep.fuzzy.var(bsetal97$ecol, bsetal97$ecol.blo)
d1 <- dudi.fca(w1, scannf = FALSE, nf = 3)
d2 <- dudi.fca(w2, scannf = FALSE, nf = 3)
plot(coinertia(d1, d2, scannf = FALSE))

## End(Not run)

### dudi.hillsmith

**Ordination of Tables mixing quantitative variables and factors**

**Description**

performs a multivariate analysis with mixed quantitative variables and factors.

**Usage**

```
dudi.hillsmith(df, row.w = rep(1, nrow(df))/nrow(df),
    scannf = TRUE, nf = 2)
```

**Arguments**

- `df`: a data frame with mixed type variables (quantitative and factor)
- `row.w`: a vector of row weights, by default uniform row weights are used
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: if scannf FALSE, an integer indicating the number of kept axes

**Details**

If `df` contains only quantitative variables, this is equivalent to a normed PCA.
If `df` contains only factors, this is equivalent to a MCA.

This analysis is the Hill and Smith method and is very similar to `dudi.mix` function. The differences are that `dudi.hillsmith` allow to use various row weights, while `dudi.mix` deals with ordered variables.
The principal components of this analysis are centered and normed vectors maximizing the sum of:
- squared correlation coefficients with quantitative variables
- correlation ratios with factors

Value

Returns a list of class mix and dudi (see dudi) containing also:

- `index`: a factor giving the type of each variable: `f = factor, q = quantitative`
- `assign`: a factor indicating the initial variable for each column of the transformed table
- `cr`: a data frame giving for each variable and each score:
  - the squared correlation coefficients if it is a quantitative variable
  - the correlation ratios if it is a factor

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


See Also

dudi.mix

Examples

data(dunedata)
attributes(dunedata$envir$use)$class <- "factor"  # use dudi.mix for ordered data
dd1 <- dudi.hillsmith(dunedata$envir, scann = FALSE)
if(adegraphicsLoaded()) {
  g <- scatter(dd1, row.plab.cex = 1, col.plab.cex = 1.5)
} else {
  scatter(dd1, clab.r = 1, clab.c = 1.5)
}
dudi.mix

Ordination of Tables mixing quantitative variables and factors

Description

performs a multivariate analysis with mixed quantitative variables and factors.

Usage

dudi.mix(df, add.square = FALSE, scannf = TRUE, nf = 2)

Arguments

df a data frame with mixed type variables (quantitative, factor and ordered)
add.square a logical value indicating whether the squares of quantitative variables should be added
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes

Details

If df contains only quantitative variables, this is equivalent to a normed PCA.
If df contains only factors, this is equivalent to a MCA.
Ordered factors are replaced by poly(x, deg=2).

This analysis generalizes the Hill and Smith method.
The principal components of this analysis are centered and normed vectors maximizing the sum of
the:
squared correlation coefficients with quantitative variables
squared multiple correlation coefficients with polynoms
correlation ratios with factors.

Value

Returns a list of class mix and dudi (see dudi) containing also

index a factor giving the type of each variable : f = factor, o = ordered, q = quantitative
assign a factor indicating the initial variable for each column of the transformed table
cr a data frame giving for each variable and each score:
the squared correlation coefficients if it is a quantitative variable
the correlation ratios if it is a factor
the squared multiple correlation coefficients if it is ordered
**dudi.nsc**

*Non symmetric correspondence analysis*

**Description**

performs a non symmetric correspondence analysis.

**Usage**

```r
dudi.nsc(df, scannf = TRUE, nf = 2)
```

**Arguments**

- `df` - a data frame containing positive or null values
- `scannf` - a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf` - if scannf FALSE, an integer indicating the number of kept axes
**Value**

Returns a list of class `nsc` and `dudi` (see `dudi`) containing also

N

sum of the values of the initial table

**Author(s)**

Daniel Chessel

Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

**References**


**Examples**

data(housetasks)
nsc1 <- dudi.nsc(housetasks, scan = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.label(nsc1$c1, plab.cex = 1.25)
  g2 <- s.arrow(nsc1$li, add = TRUE, plab.cex = 0.75)
} else {
  s.label(nsc1$c1, clab = 1.25)
  s.arrow(nsc1$li, add.pl = TRUE, clab = 0.75) # see ref p.383
}

---

dudi.pca

**Principal Component Analysis**

**Description**

dudi.pca performs a principal component analysis of a data frame and returns the results as objects of class `pca` and `dudi`.

**Usage**

dudi.pca(df, row.w = rep(1, nrow(df))/nrow(df),
          col.w = rep(1, ncol(df)), center = TRUE, scale = TRUE,
          scannf = TRUE, nf = 2)

**Arguments**

df a data frame with n rows (individuals) and p columns (numeric variables)

row.w an optional row weights (by default, uniform row weights)

col.w an optional column weights (by default, unit column weights)
center  a logical or numeric value, centring option
if TRUE, centring by the mean
if FALSE no centring
if a numeric vector, its length must be equal to the number of columns of the
data frame df and gives the centring

scale   a logical value indicating whether the column vectors should be normed for the
row.w weighting

scannf  a logical value indicating whether the screeplot should be displayed

nf      if scannf FALSE, an integer indicating the number of kept axes

Value

Returns a list of classes pca and dudi (see dudi) containing the used information for computing the
principal component analysis :

  tab    the data frame to be analyzed depending of the transformation arguments (center
          and scale)
  cw     the column weights
  lw     the row weights
  eig    the eigenvalues
  rank   the rank of the analyzed matrice
  nf     the number of kept factors
  c1     the column normed scores i.e. the principal axes
  l1     the row normed scores
  co     the column coordinates
  li     the row coordinates i.e. the principal components
  call   the call function
  cent   the p vector containing the means for variables (Note that if center = F, the
          vector contains p 0)
  norm   the p vector containing the standard deviations for variables i.e. the root of the
          sum of squares deviations of the values from their means divided by n (Note that
          if norm = F, the vector contains p 1)

Author(s)

  Daniel Chessel
  Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

See Also

  prcomp, princomp in the mva library
Examples

data(deug)
deg.dudi <- dudi.pco(deug$tab, center = deug$cent, scale = FALSE, scan = FALSE)
deg.dudi1 <- dudi.pca(deug$tab, center = TRUE, scale = TRUE, scan = FALSE)

if(adegraphicsLoaded()) {
g1 <- s.class(deug.dudi$li, deug$result, plot = FALSE)
g2 <- s.arrow(deug.dudi$c1, lab = names(deug$tab), plot = FALSE)
g3 <- s.class(deug.dudi1$li, deug$result, plot = FALSE)
g4 <- s.corcircle(deug.dudi1$co, lab = names(deug$tab), full = FALSE, plot = FALSE)
G1 <- rbindADEg(cbindADEg(g1, g2, plot = FALSE), cbindADEg(g3, g4, plot = FALSE), plot = TRUE)
G2 <- s1d.hist(deug.dudi$tab, breaks = seq(-45, 35, by = 5), type = "density", xlim = c(-40, 40),
             right = FALSE, ylim = c(0, 0.1), porigin.lwd = 2)
} else {
  par(mfrow = c(2, 2))
s.class(deug.dudi$li, deug$result, cpoint = 1)
s.arrow(deug.dudi$c1, lab = names(deug$tab))
s.class(deug.dudi1$li, deug$result, cpoint = 1)
s.corcircle(deug.dudi1$co, lab = names(deug$tab), full = FALSE, box = TRUE)
  par(mfrow = c(1, 1))
}

# for interpretations
par(mfrow = c(3, 3))
par(mar = c(2.1, 2.1, 2.1, 1.1))
for(i in 1:9) {
  hist(deug.dudi$tab[,i], xlim = c(-40, 40), breaks = seq(-45, 35, by = 5),
       prob = TRUE, right = FALSE, main = names(deug$tab)[i], xlab = "", ylim = c(0, 0.1))
  abline(v = 0, lwd = 3)
}
  par(mfrow = c(1, 1))
}

---

**dudi.pco**

**Principal Coordinates Analysis**

**Description**

*dudi.pco* performs a principal coordinates analysis of a Euclidean distance matrix and returns the results as objects of class *pco* and *dudi*.

**Usage**

```r
?dudi.pco(d, row.w = "uniform", scannf = TRUE, nf = 2,
          full = FALSE, tol = 1e-07)
```

## S3 method for class 'pco'

```r
scatter(x, xax = 1, yax = 2, clab.row = 1, posieig = "top",
       sub = NULL, csub = 2, ...)
```
Arguments

- **d**: an object of class `dist` containing a Euclidean distance matrix.
- **row.w**: an optional distance matrix row weights. If not NULL, must be a vector of positive numbers with length equal to the size of the distance matrix.
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed.
- **nf**: if scannf FALSE, an integer indicating the number of kept axes.
- **full**: a logical value indicating whether all the axes should be kept.
- **tol**: a tolerance threshold to test whether the distance matrix is Euclidean: an eigenvalue is considered positive if it is larger than \(-tol*\lambda_1\) where \(\lambda_1\) is the largest eigenvalue.

- **x**: an object of class `pco`.
- **xax**: the column number for the x-axis.
- **yax**: the column number for the y-axis.
- **clab.row**: a character size for the row labels.
- **posieig**: if "top" the eigenvalues bar plot is upside, if "bottom" it is downside, if "none" no plot.
- **sub**: a string of characters to be inserted as legend.
- **csub**: a character size for the legend, used with `par("cex")*csub`.
- **...**: further arguments passed to or from other methods.

Value

dudi.pco returns a list of class `pco` and `dudi`. See `dudi`.

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(yanomama)
gen <- quasieuclid(as.dist(yanomama$gen))
geo <- quasieuclid(as.dist(yanomama$geo))
ant <- quasieuclid(as.dist(yanomama$ant))
geo1 <- dudi.pco(geo, scann = FALSE, nf = 3)
gen1 <- dudi.pco(gen, scann = FALSE, nf = 3)
ant1 <- dudi.pco(ant, scann = FALSE, nf = 3)
plot(coinertia(ant1, gen1, scann = FALSE))
dunedata  

*Dune Meadow Data*

**Description**

dunedata is a data set containing for 20 sites, environmental variables and plant species.

**Usage**

data(dunedata)

**Format**

dunedata is a list with 2 components.

- **envir** is a data frame with 20 rows (sites) 5 columns (environmental variables).
- **veg** is a data frame with 20 rows (sites) 30 columns (plant species).

**Source**


**Examples**

data(dunedata)
summary(dunedata$envir)
is.ordered(dunedata$envir$use)
score(dudi.mix(dunedata$envir, scan = FALSE))

ecg  

*Electrocardiogram data*

**Description**

These data were measured during the normal sinus rhythm of a patient who occasionally experiences arrhythmia. There are 2048 observations measured in units of millivolts and collected at a rate of 180 samples per second. This time series is a good candidate for a multiresolution analysis because its components are on different scales. For example, the large scale (low frequency) fluctuations, known as baseline drift, are due to the patient respiration, while the prominent short scale (high frequency) intermittent fluctuations between 3 and 4 seconds are evidently due to patient movement. Heart rhythm determines most of the remaining features in the series. The large spikes occurring about 0.7 seconds apart the R waves of normal heart rhythm; the smaller, but sharp peak coming just prior to an R wave is known as a P wave; and the broader peak that comes after a R wave is a T wave.
ecomor

Usage

data(ecg)

Format

A vector of class ts containing 2048 observations.

Source

Gust Bardy and Per Reinhall, University of Washington

References


Examples

```r
## Not run:
# figure 130 in Percival and Walden (2000)
if (requireNamespace("waveslim") == TRUE) {
  data(ecg)
  ecg.level <- haar2level(ecg)
  ecg.haar <- orthobasis.haar(length(ecg))
  ecg.mld <- mld(ecg, ecg.haar, ecg.level, plot = FALSE)
  res <- cbind.data.frame(apply(ecg.mld[,1:5],1,sum), ecg.mld[,6:11])
  par(mfrow = c(8,1))
  par(mar = c(2, 5, 1.5, 0.6))
  plot(as.ts(ecg), ylab = "ECG")
  apply(res, 2, function(x) plot(as.ts(x), ylim = range(res),
                               ylab = ""))
  par(mfrow = c(1,1))
}
## End(Not run)
```

describe(ecomor)

Ecomorphological Convergence

description

This data set gives ecomorphological informations about 129 bird species.

Usage

data(ecomor)
Format

ecomor is a list of 7 components.

forsub is a data frame with 129 species, 6 variables (the feeding place classes): foliage, ground, twig, bush, trunk and aerial feeders. These dummy variables indicate the use (1) or no use (0) of a given feeding place by a species.

diet is a data frame with 129 species and 8 variables (diet types): Gr (granivorous: seeds), Fr (frugivorous: berries, acorns, drupes), Ne (frugivorous: nectar), Fo (folivorous: leaves), In (invertebrate feeder: insects, spiders, myriapods, isopods, snails, worms), Ca (carnivorous: flesh of small vertebrates), Li (limnivorous: invertebrates in fresh water), and Ch (carrion feeder). These dummy variables indicate the use (1) or no use (0) of a given diet type by a species.

habitat is a data frame with 129 species, 16 dummy variables (the habitats). These variables indicate the species presence (1) or the species absence (0) in a given habitat.

morpho is a data frame with 129 species abd 8 morphological variables: wingl (Wing length, mm), taill (Tail length, mm), culml (Culmen length, mm), bilh (Bill height, mm), bilw (Bill width, mm), tarsl (Tarsus length, mm), midtl (Middle toe length, mm) and weig (Weight, g).

taxo is a data frame with 129 species and 3 factors: Genus, Family and Order. It is a data frame of class `taxo`: the variables are factors giving nested classifications.

labels is a data frame with vectors of the names of species (complete and in abbreviated form.

categ is a data frame with 129 species, 2 factors: 'forsub' summarizing the feeding place and 'diet' the diet type.

Source


References


Examples

data(ecomor)
$r<-$ apply(ecomor$habitat, 2, sum)
$s.corcircle(dudi.pca(log(ecomor$morpho), scan = FALSE)$co)

forsub <- data.frame(t(apply(ecomor$forsub, 1, function (x) x / sum(x))))
$pca1$ <- dudi.pca(forsub, scan = FALSE, scale = FALSE)
$w1$ <- as.matrix(forsub)
if(adegraphicsLoaded()) {
  $g1$ <- s.arrow(pca1$c1, plot = FALSE)
  $g2$ <- s.label(w1, plab.cex = 0, ppoi.cex = 2, plot = FALSE)
  $G1$ <- superpose($g1$, $g2$, plot = TRUE)
} else {
**elec88**  

**Electoral Data**

**Description**

This data set gives the results of the presidential election in France in 1988 for each department and all the candidates.

**Usage**

```r
data(elec88)
```
Format

elec88 is a list with the following components:

- **tab** a data frame with 94 rows (departments) and 9 variables (candidates)
- **res** the global result of the election all-over the country
- **lab** a data frame with two variables: elec88$lab$dep is a vector containing the names of the 94 french departments, elec88$lab$reg is a vector containing the names of the 21 French administrative regions.
- **area** the data frame of 3 variables returning the boundary lines of each department. The first variable is a factor. The levels of this one are the row.names of tab. The second and third variables return the coordinates (x, y) of the points of the boundary line.
- **contour** a data frame with 4 variables (x1, y1, x2, y2) for the contour display of France
- **xy** a data frame with two variables (x, y) giving the position of the center for each department
- **neig** the neighbouring graph between departments, object of the class neig
- **nb** the neighbouring graph between departments, object of the class nb
- **Spatial** the map of the french departments in Lambert II coordinates (an object of the class SpatialPolygons of sp)
- **Spatial.contour** the contour of the map of France in Lambert II coordinates (an object of the class SpatialPolygons of sp)

Source

Public data

See Also

This dataset is compatible with presid2002 and cnc2003

Examples

data(elec88)
apply(elec88$tab, 2, mean)
summary(elec88$res)
pca1 <- dudi.pca(elec88$tab, scale = FALSE, scannf = FALSE)

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    data1 <- as.data.frame(as.numeric(rownames(elec88$tab) == "D25"))
    rownames(data1) <- row.names(elec88$Spatial)
    obj1 <- sp::SpatialPolygonsDataFrame(Sr = elec88$Spatial, data = data1)
    g1 <- s.Spatial(obj1, psub.text = "", plot = FALSE)
    g2 <- s.Spatial(obj1, psub.text = "", nb = elec88$nb, pnb.node.cex = 0, plot = FALSE)

    data3 <- as.data.frame(elec88$xy[, 1] + elec88$xy[, 2])
    rownames(data3) <- row.names(elec88$Spatial)
    obj3 <- sp::SpatialPolygonsDataFrame(Sr = elec88$Spatial, data = data3)
    g3 <- s.Spatial(obj3, psub.text = "", plot = FALSE)
data4 <- as.data.frame(pca1$li[, 1])
rownames(data4) <- row.names(elec88$Spatial)
obj4 <- sp::SpatialPolygonsDataFrame(Sr = elec88$Spatial, data = data4)
g4 <- s.Spatial(obj4, psub.text = "F1 PCA", plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
}
} else {
  par(mfrow = c(2, 2))
  plot(elec88$area[, 2:3], type = "n", asp = 1)
  lpoly <- split(elec88$area[, 2:3], elec88$area[, 1])
  lapply(lpoly, function(x) {points(x, type = "l"); invisible()})
  polygon(elec88$area[elec88$area$V1 == "D25", 2:3], col = 1)
  area.plot(elec88$area, graph = elec88$neig, lwdg = 1)
  polygon(elec88$area[elec88$area$V1 == "D25", 2:3], col = 1)
  area.plot(elec88$area, val = elec88$xy[, 1] + elec88$xy[, 2])
  area.plot(elec88$area, val = pca1$li[, 1], sub = "F1 PCA",
            csub = 2, cleg = 1.5)
  par(mfrow = c(1, 1))
}

---

escopage

**K-tables of wine-tasting**

**Description**

This data set describes 27 characteristics of 21 wines distributed in four fields: rest, visual, olfactory and global.

**Usage**

data(escopage)

**Format**

escopage is a list of 3 components.

- **tab** is a data frame with 21 observations (wines) and 27 variables.
- **tab.names** is the vector of the names of sub-tables: "rest" "visual" "olfactory" "global".
- **blo** is a vector of the numbers of variables for each sub-table.

**Source**


Examples

```r
data(escopage)
w <- data.frame(scale(escopage$tab))
w <- ktab.data.frame(w, escopage$blo)
names(w)[1:4] <- escopage$tab.names
plot(mfa(w, scan = FALSE))
```

---

**euro123**

*Triangular Data*

Description

This data set gives the proportions of employment in the primary, secondary and tertiary sectors for 12 European countries in 1978, 1986 and 1997.

Usage

```r
data(euro123)
```

Format

euro123 is a list of 4 components.

- **in78**: a data frame with 12 rows and 3 variables.
- **in86**: idem in 1986
- **in97**: idem in 1997
- **plan**: is a data frame with two factors to both organize the 3 tables.

Source


Examples

```r
data(euro123)
if(adeGraphicsLoaded()) {
  g1 <- triangle.label(euro123$in78, addaxes = TRUE, plabels.cex = 0,
      plot = FALSE)
  g2 <- triangle.label(euro123$in86, addaxes = TRUE, plabels.cex = 0,
      plot = FALSE)
  g3 <- triangle.label(euro123$in97, addaxes = TRUE, plabels.cex = 0,
      plot = FALSE)
  g4 <- triangle.match(euro123$in78, euro123$in97, plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
}
```
fission

Fission pattern and heritable morphological traits

Description

This data set contains the mean values of five highly heritable linear combinations of cranial metric (GM1-GM3) and non metric (GN1-GN2) for 8 social groups of Rhesus Macaques on Cayo Santiago. It also describes the fission tree depicting the historical phyletic relationships.

Usage

data(fission)

Format

fission is a list containing the 2 following objects:

- tre is a character string giving the fission tree in Newick format.
- tab is a data frame with 8 social groups and five traits: cranial metrics (GM1, GM2, GM3) and cranial non metrics (GN1, GN2)

References


Examples

data(fission)
fis.phy <- newick2phylog(fission$tre)
table.phylog(fission$tab[, names(fis.phy$leaves)], fis.phy, csi = 2)
gearymoran(fis.phy$Amat, fission$tab)
Description

K tables have the same rows and the same columns. Each table is transformed by \( P = X/\text{sum}(X) \). The average of \( P \) is computing. A correspondence analysis is realized on this average. The initial rows and the initial columns are projected in supplementary elements.

Usage

```r
foucart(X, scannf = TRUE, nf = 2)
## S3 method for class 'foucart'
plot(x, xax = 1, yax = 2, clab = 1, csub = 2,
    possub = "bottomright", ...)
## S3 method for class 'foucart'
print(x, ...)
```

Arguments

- **X**: a list of data frame where the row names and the column names are the same for each table
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes
- **x**: an object of class 'foucart'
- **xax**: the column number of the x-axis
- **yax**: the column number of the y-axis
- **clab**: if not NULL, a character size for the labels, used with `par("cex")*clab`
- **csub**: a character size for the legend, used with `par("cex")*csub`
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- **...**: further arguments passed to or from other methods

Value

`foucart` returns a list of the classes 'dudi', 'coa' and 'foucart'

```r
call           origine
nf            axes-components saved
rank          rank
blo           useful vector
```
cw vector: column weights
lw vector: row weights
eig vector: eigen values
tab data.frame: modified array
li data.frame: row coordinates
ll data.frame: row normed scores
co data.frame: column coordinates
c1 data.frame: column normed scores
Tli data.frame: row coordinates (each table)
Tco data.frame: col coordinates (each table)
TL data.frame: factors for Tli
TC data.frame: factors for Tco

Author(s)
Pierre Bady <pierre.bady@univ-lyon1.fr>
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References

Examples
data(bf88)
fou1 <- foucart(bf88, scann = FALSE, nf = 3)
fou1
plot(fou1)

data(meaudret)
l1 <- split(meaudret$spe, meaudret$design$season)
l1 <- lapply(l1, function(x)
{row.names(x) <- paste("Sit",1:5,sep="");x})
fou2 <- foucart(l1, scan = FALSE)

if(adegraphicsLoaded()) {
  kplot(fou2, row.plabels.cex = 2)
} else {
  kplot(fou2, clab.r = 2)
}
Description

These functions allow to compute the fourth-corner statistic for abundance or presence-absence data. The fourth-corner statistic has been developed by Legendre et al (1997) and extended in Dray and Legendre (2008). The statistic measures the link between three tables: a table L (n x p) containing the abundances of p species at n sites, a second table R (n x m) containing the measurements of m environmental variables for the n sites, and a third table Q (p x s) describing s species traits for the p species.

Usage

fourthcorner2(tabR, tabL, tabQ, modeltype = 6, nrepet = 999, tr01 = FALSE, p.adjust.method.G = p.adjust.methods, ...)  
## S3 method for class '4thcorner'
print(x, varQ = 1:length(x$varnames.Q), varR = 1:length(x$varnames.R), stat = c("D", "D2"), ...)  
## S3 method for class '4thcorner'
summary(object,...)  
## S3 method for class '4thcorner'
plot(x, stat = c("D", "D2", "G"), type = c("table", "biplot"), xax = 1, yax = 2, x.rlq = NULL, alpha = 0.05, col = c("lightgrey", "red", "deepskyblue", "purple"), ...)  

Arguments

tabR a dataframe containing the measurements (numeric values or factors) of m environmental variables (columns) for the n sites (rows).
tabL a dataframe containing the abundances of p species (columns) at n sites (rows).
tabQ a dataframe containing numeric values or factors describing s species traits (columns) for the p species (rows).
modeltype an integer (1-6) indicating the permutation model used in the testing procedure (see details).
nrepet the number of permutations
tr01 a logical indicating if data in tabL must be transformed to presence-absence data (FALSE by default)
object an object of the class fourthcorner
x an object of the class fourthcorner
varR a vector containing indices for variables in tabR
varQ a vector containing indices for variables in tabQ
type results are represented by a table or on a biplot (see x.rlq)
alpha a value of significance level
p.adjust.method.G a string indicating a method for multiple adjustment used for output tabG, see p.adjust.methods for possible choices
p.adjust.method.D a string indicating a method for multiple adjustment used for output tabD/tabD2, see p.adjust.methods for possible choices
p.adjust.D a string indicating if multiple adjustment for tabD/tabD2 should be done globally or only between levels of a factor ("levels", as in the original paper of Legendre et al. 1997)
stat a character to specify if results should be plotted for cells (D and D2) or variables (G)
xax an integer indicating which rlq axis should be plotted on the x-axis
yax an integer indicating which rlq axis should be plotted on the y-axis
x.rlq an object created by the rlq function. Used to represent results on a biplot (type should be "biplot" and object created by the fourthcorner functions)
col a vector of length 4 containing four colors used for the graphical representations. The first is used to represent non-significant associations, the second positive significant, the third negative significant. For the 'biplot' method and objects created by the fourthcorner.rlq function, the second corresponds to variables significantly linked to the x-axis, the third for the y-axis and the fourth for both axes
xtest an object created by the rlq function
typetest a string indicating which tests should be performed
... further arguments passed to or from other methods

Details

For the fourthcorner function, the link is measured by a Pearson correlation coefficient for two quantitative variables (trait and environmental variable), by a Pearson Chi2 and G statistic for two qualitative variables and by a Pseudo-F and Pearson r for one quantitative variable and one qualitative variable. The fourthcorner2 function offers a multivariate statistic (equal to the sum of eigenvalues of RLQ analysis) and measures the link between two variables by a square correlation coefficient (quant/quant), a Chi2/sum(L) (qual/qual) and a correlation ratio (quant/qual). The significance is tested by a permutation procedure. Different models are available:

- model 1 (modeltype=1): Permute values for each species independently (i.e., permute within each column of table L)
- model 2 (modeltype=2): Permute values of sites (i.e., permute entire rows of table L)
model 3 (modeltype=3): Permute values for each site independently (i.e., permute within each row of table L)
model 4 (modeltype=4): Permute values of species (i.e., permute entire columns of table L)
model 5 (modeltype=5): Permute values of species and after (or before) permute values of sites (i.e., permute entire columns and after (or before) entire rows of table L)
model 6 (modeltype=6): combination of the outputs of models 2 and 4. Dray and Legendre (2008) and ter Braak et al. (20012) showed that all models (except model 6) have inflated type I error.

Note that the model 5 is strictly equivalent to permuting simultaneously the rows of tables R and Q, as proposed by Doledec et al. (1996).
The function summary returns results for variables (G). The function print returns results for cells (D and D2). In the case of qualitative variables, Holm’s corrected pvalues are also provided.
The function plot produces a graphical representation of the results (white for non significant, light grey for negative significant and dark grey for positive significant relationships). Results can be plotted for variables (G) or for cells (D and D2). In the case of qualitative / quantitative association, homogeneity (D) or correlation (D2) are plotted.

Value
The fourthcorner function returns a a list where:
- tabD is a krandtest object giving the results of tests for cells of the fourth-corner (homogeneity for quant./qual.).
- tabD2 is a krandtest object giving the results of tests for cells of the fourth-corner (Pearson r for quant./qual.).
- tabG is a krandtest object giving the results of tests for variables (Pearson’s Chi2 for qual./qual.).

The fourthcorner2 function returns a list where:
- tabG is a krandtest object giving the results of tests for variables.
- trRLQ is a krandtest object giving the results of tests for the multivariate statistic (i.e. equivalent to randtest.rlq function).

Author(s)
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References
### Examples

```r
data(aviurba)

## Version using the sequential test (ter Braak et al 2012)
## as recommended in Dray et al (2013),
## using Holm correction of P-values (only 99 permutations here)
four.comb.default <- fourthcorner(aviurba$mil, aviurba$fau, aviurba$traits, nrepet=99)
summary(four.comb.default)
plot(four.comb.default, stat = "G")

## using fdr correction of P-values
four.comb.fdr <- fourthcorner(aviurba$mil, aviurba$fau, aviurba$traits,
summary(four.comb.fdr)
plot(four.comb.fdr, stat = "G")

## Explicit procedure to combine the results of two models
## proposed in Dray and Legendre (2008); the above does this implicitly
four2 <- fourthcorner(aviurba$mil, aviurba$fau, aviurba$traits, nrepet=99, modeltype=2)
four4 <- fourthcorner(aviurba$mil, aviurba$fau, aviurba$traits, nrepet=99, modeltype=4)
four.comb <- combine.4thcorner(four2, four4)
summary(four.comb)
plot(four.comb, stat = "G")
```

---

**friday87**

*Faunistic K-tables*

### Description

This data set gives informations about sites, species and environmental variables.

### Usage

```r
data(friday87)
```

### Format

*friday87* is a list of 4 components.

- **fau** is a data frame containing a faunistic table with 16 sites and 91 species.
- **mil** is a data frame with 16 sites and 11 environmental variables.
- **fau.blo** is a vector of the number of species per group.
- **tab.names** is the name of each group of species.
Source

Examples

data(friday87)
wfri <- data.frame(scale(friday87$fau, scal = FALSE))
wfri <- ktab.data.frame(wfri, friday87$fau.blo,
    tabnames = friday87$tab.names)

if(adegraphicsLoaded()) {
    g1 <- kplot(sepan(wfri), row.plabels.cex = 2)
} else {
    kplot(sepan(wfri), clab.r = 2, clab.c = 1)
}

---

fruits  
*Pair of Tables*

Description
28 batches of fruits -two types- are judged by two different ways. They are classified in order of preference, without ex aequo, by 16 individuals. 15 quantitative variables described the batches of fruits.

Usage

data(fruits)

Format
fruits is a list of 3 components:
  *typ* is a vector returning the type of the 28 batches of fruits (peaches or nectarines).
  *jug* is a data frame of 28 rows and 16 columns (judges).
  *var* is a data frame of 28 rows and 16 measures (average of 2 judgements).

Details
fruits$var is a data frame of 15 variables:
1. taches: quantity of cork blemishes (0=absent - maximum 5)
2. stries: quantity of stria (1/none - maximum 4)
3. abmucr: abundance of mucron (1/absent - 4)
4. inform: shape irregularity (0/none - 3)
fruits

5. allong: length of the fruit (1/round fruit - 4)
6. suroug: percentage of the red surface (minimum 40% - maximum 90%)
7. homlot: homogeneity of the intra-batch coloring (1/strong - 4)
8. homfru: homogeneity of the intra-fruit coloring (1/strong - 4)
9. pubesc: pubescence (0/none - 4)
10. verrou: intensity of green in red area (1/none - 4)
11. foncee: intensity of dark area (0/pink - 4)
12. comucr: intensity of the mucron color (1=no contrast - 4/dark)
13. impres: kind of impression (1/watched - 4/pointillé)
14. coldom: intensity of the predominating color (0/clear - 4)
15. calibr: grade (1/<90g - 5/>200g)

Source

Examples

data(fruits)
pcajug <- dudi.pca(fruits$jug, scann = FALSE)
pcavar <- dudi.pca(fruits$var, scann = FALSE)

if(adegraphicsLoaded()) {
  g1 <- s.corcircle(pcajug$co, plot = FALSE)
  g2 <- s.class(pcajug$li, fac = fruits$type, plot = FALSE)
  g3 <- s.corcircle(pcavar$co, plot = FALSE)
  g4 <- s.class(pcavar$li, fac = fruits$type, plot = FALSE)

  G1 <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
  G2 <- plot(coinertia(pcajug, pcavar, scan = FALSE))
} else {
  par(mfrow = c(2,2))
  s.corcircle(pcajug$co)
  s.class(pcajug$li, fac = fruits$type)
  s.corcircle(pcavar$co)
  s.class(pcavar$li, fac = fruits$type)

  par(mfrow = c(1,1))
  plot(coinertia(pcajug, pcavar, scan = FALSE))
}
### gearymoran

**Moran's I and Geary's c randomization tests for spatial and phylogenetic autocorrelation**

**Description**

This function performs Moran's I test using phylogenetic and spatial link matrix (binary or general).

It uses neighbouring weights so Moran's I and Geary's c randomization tests are equivalent.

**Usage**

```r
gearymoran(bilis, X, nrepet = 999, alter = c("greater", "less", "two-sided"))
```

**Arguments**

- `bilis`: a \( n \) by \( n \) link matrix where \( n \) is the row number of \( X \)
- `X`: a data frame with continuous variables
- `nrepet`: number of random vectors for the randomization test
- `alter`: a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two-sided"

**Details**

- \( \text{bilis} \) is a squared symmetric matrix which terms are all positive or null.
- \( \text{bilis} \) is firstly transformed in frequency matrix \( A \) by dividing it by the total sum of data matrix:

\[
a_{ij} = \frac{\text{bilis}_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} \text{bilis}_{ij}}
\]

The neighbouring weights is defined by the matrix \( D = \text{diag}(d_1, d_2, \ldots) \) where \( d_i = \sum_{j=1}^{n} \text{bilis}_{ij} \).

For each vector \( x \) of the data frame \( X \), the test is based on the Moran statistic \( x'Ax \) where \( x \) is D-centred.

**Value**

Returns an object of class `krandtest` (randomization tests).

**Author(s)**

Sébastien Ollier <sebastien.ollier@u-psud.fr>
Daniel Chessel

**References**


See Also

*moran.test* and *geary.test* for classical versions of Moran’s test and Geary’s one

Examples

```r
# a spatial example
data(mafragh)
tab0 <- as.data.frame(scalewt(mafragh$env))
bilis0 <- neig2mat(mafragh$neig)
gm0 <- gearymoran(bilis0, tab0, 999)
gm0
plot(gm0, nclass = 20)

## Not run:
# a phylogenetic example
data(mjrochet)
mjr.phy <- newick2phylog(mjrochet$tre)
mjr.tab <- log(mjrochet$tab)
gearymoran(mjr.phy$Amat, mjr.tab)
gearymoran(mjr.phy$Wmat, mjr.tab)

if(adegraphicsLoaded()) {
  g1 <- table.value(mjr.phy$Wmat, ppoints.cex = 0.35, nclass = 5,
                    axis.text = list(cex = 0), plot = FALSE)
  g2 <- table.value(mjr.phy$Amat, ppoints.cex = 0.35, nclass = 5,
                    axis.text = list(cex = 0), plot = FALSE)
  G <- cbindADEg(g1, g2, plot = TRUE)
} else {
  par(mfrow = c(1, 2))
  table.value(mjr.phy$Wmat, csi = 0.25, clabel.r = 0)
  table.value(mjr.phy$Amat, csi = 0.35, clabel.r = 0)
  par(mfrow = c(1, 1))
}

## End(Not run)
```

## ggtortoises

*Microsatellites of Galapagos tortoises populations*

Description

This data set gives genetic relationships between Galapagos tortoises populations with 10 microsatellites.

Usage

data(ggtortoises)
ggtortoises is a list with the following components:

- **area**: a data frame designed to be used in the area.plot function
- **ico**: a list of three pixmap icons representing the tortoises morphotypes
- **pop**: a data frame containing meta informations about populations
- **misc**: a data frame containing the coordinates of the island labels
- **loc**: a numeric vector giving the number of alleles by marker
- **tab**: a data frame containing the number of alleles by populations for 10 microsatellites
- **Spatial**: an object of the class SpatialPolygons of sp, containing the map

**Source**


**References**


**Examples**

```r
if(requireNamespace("pixmap", quietly=TRUE)) {
  data(ggtortoises)

  if(adegraphicsLoaded()) {
    if(requireNamespace("sp", quietly = TRUE)) {
      g1 <- s.logo(ggtortoises$pop, ggtortoises$ico[as.character(ggtortoises$pop$carap)],
        Sp = ggtortoises$Spatial, pbackground.col = "lightblue", pSp.col = "white",
        pgrid.draw = FALSE, ppoints.cex = 0.5)
      g1 <- s.label(ggtortoises$misc, pgrid.draw = FALSE, porigin.include = FALSE,
        paxes.draw = FALSE, add = TRUE)
    }
  } else {
    a1 <- ggtortoises$area
    area.plot(a1)
    rect(min(a1$x), min(a1$y), max(a1$x), max(a1$y), col = "lightblue")
    invisible(lapply(split(a1, a1$id), function(x) polygon(x[, -1], col = "white")))
    s.label(ggtortoises$misc, grid = FALSE, include.ori = FALSE, addaxies = FALSE, add.p = TRUE)
    listico <- ggtortoises$ico[as.character(ggtortoises$pop$carap)]
    s.logo(ggtortoises$pop, listico, add.p = TRUE)
  }
}
```
granulo  Granulometric Curves

Description

This data set gives the repartition in diameter classes of deposit samples.

Usage

data(granulo)

Format

granulo is a list of 2 components.

- `tab` contains the 49 deposit samples, 9 diameter classes, weight of grains by size class
- `born` contains the boundaries of the diameter classes

Source


Examples

data(granulo)
w <- t(apply(granulo$tab, 1, function (x) x / sum(x)))
w <- data.frame(w)
wtr <- data.frame(t(w))
wmoy <- data.frame(matrix(apply(wtr, 1, mean), 1))
d1 <- dudi.pca(w, scal = FALSE, scan = FALSE)
wmoy <- suprow(d1, wmoy)$lisup

if(aderaphicsLoaded()) {
s.arrow(d1$e1, plab.cex = 1.5)
s.distri(d1$e1, wtr, starSize = 0.33, ellipseSize = 0,
        add = TRUE, plab.cex = 0.75)
s.label(wmoy, ppoints.cex = 5, plab.cex = 0, add = TRUE)
} else {
    s.arrow(d1$e1, clab = 1.5)
s.distri(d1$e1, wtr, cstar = 0.33, cell = 0,
        axesell = FALSE, add.p = TRUE, clab = 0.75)
s.label(wmoy, cpoi = 5, clab = 0, add.p = TRUE)
}
### gridrowcol

**Complete regular grid analysis**

**Description**

This function defines objects to analyse data sets associated with complete regular grid.

**Usage**

```r
gridrowcol(nrow, ncol, cell.names = NULL)
```

**Arguments**

- `nrow` : size of the grid (number of rows)
- `ncol` : size of the grid (number of columns)
- `cell.names` : grid cell labels

**Value**

Returns a list containing the following items :

- `xy` : a data frame with grid cell coordinates
- `area` : a data frame with three variables to display grid cells as areas
- `neig` : an object of class 'neig' corresponding to a neighbouring graph of the grid (rook case)
- `orthobasis` : an object of class 'orthobasis' corresponding to the analytical solution for the neighbouring graph

**Author(s)**

Sébastien Ollier <sebastien.ollier@u-psud.fr>
Daniel Chessel

**References**


**See Also**

- `orthobasis`
- `orthogram`
- `mld`
Examples

```r
w <- gridrowcol(8, 5)
par(mfrow = c(1, 2))
area.plot(w$area, center = w$xy, graph = w$neig, clab = 0.75)
area.plot(w$area, center = w$xy, graph = w$neig, clab = 0.75, label = as.character(1:40))
par(mfrow = c(1, 1))
if(adegraphicsLoaded()) {
  fac1 <- w$orthobasis
  names(fac1) <- as.character(signif(attr(w$orthobasis, "values"), 3))
s.value(w$xy, fac1, porigin.include = FALSE, plegend.drawKey = FALSE, pgrid.text.cex = 0,
         ylim = c(0, 10))
} else {
  par(mfrow = c(5,8))
  for(k in 1:39)
    s.value(w$xy, w$orthobasis[, k], csi = 3, cleg = 0, csub = 2,
            sub = as.character(signif(attr(w$orthobasis, "values")[, k], 3)),
            incl = FALSE, addax = FALSE, cgr = 0, ylim = c(0,10))
  par(mfrow = c(1,1))
}
```

---

**hdpg**

*Genetic Variation In Human Populations*

Description

This data set gives genotypes variation of 1066 individuals belonging to 52 predefined populations, for 404 microsatellite markers.

Usage

data(hdpg)

Format

hdpg is a list of 3 components.

- **tab** is a data frame with the genotypes of 1066 individuals encoded with 6 characters (individuals in row, locus in column), for example '123098' for a heterozygote carrying alleles '123' and '098', '123123' for a homozygote carrying two alleles '123' and, '000000' for a not classified locus (missing data).

- **ind** is a a data frame with 4 columns containing information about the 1066 individuals: hdpg$ind$id containing the Diversity Panel identification number of each individual, and three factors hdpg$ind$sex, hdpg$ind$population and hdpg$ind$region containing the names of the 52 populations belonging to 7 major geographic regions (see details).
locus is a dataframe containing four columns: hdpg$locus$marknames a vector of names of the microsatellite markers, hdpg$locus$allbyloc a vector containing the number of alleles by loci, hdpg$locus$chromosome a factor defining a number for one chromosome and, hdpg$locus$maposition indicating the position of the locus in the chromosome.

Details

The rows of hdpg$pop are the names of the 52 populations belonging to the geographic regions contained in the rows of hdpg$region. The chosen regions are: America, Asia, Europe, Middle East North Africa, Oceania, Subsaharan AFRICA.

The 52 populations are: Adygei, Balochi, Bantu, Basque, Bedouin, Bergamo, Biaka Pygmies, Brahui, Burusho, Cambodian, Columbian, Dai, Daur, Druze, French, Han, Hazara, Hezhen, Japanese, Kalash, Karitiana, Lahu, Makrani, Mandenka, Maya, Mbuti Pygmies, Melanesian, Miaozu, Mongola, Mozabite, Naxi, NewGuinea, Nilote, Orcadian, Oroqen, Palestinian, Pathan, Pima, Russian, San, Sardinian, She, Sindhi, Surui, Tu, Tujiua, Tuscan, Uygur, Xibo, Yakut, Yizu, Yoruba.

hdpg$freq is a data frame with 52 rows, corresponding to the 52 populations described above, and 4992 microsatellite markers.

Source


Examples

data(hdpg)  
names(hdpg)  
str(hdpg)

________________________________________________________________________

housetasks Contingency Table

________________________________________________________________________

Description

The housetasks data frame gives 13 housetasks and their repartition in the couple.

Usage

data(housetasks)
humDNAm

Format

This data frame contains four columns: wife, alternating, husband and jointly. Each column is a numeric vector.

Source


Examples

data(housetasks)
nsc1 <- dudi.nsc(housetasks, scan = FALSE)

if(adegraphicsLoaded()) {
  s.label(nsc1$c1, plab.cex = 1.25)
  s.arrow(nsc1$li, add = TRUE, plab.cex = 0.75)
} else {
  s.label(nsc1$c1, clab = 1.25)
  s.arrow(nsc1$li, add.pl = TRUE, clab = 0.75)
}

humDNAm

*human mitochondrial DNA restriction data*

Description

This data set gives the frequencies of haplotypes of mitochondrial DNA restriction data in ten populations all over the world. It gives also distances among the haplotypes.

Usage

data(humDNAm)

Format

humDNAm is a list of 3 components.

- **distances** is an object of class `dist` with 56 haplotypes. These distances are computed by counting the number of differences in restriction sites between two haplotypes.

- **samples** is a data frame with 56 haplotypes, 10 abundance variables (populations). These variables give the haplotype abundance in a given population.

- **structures** is a data frame with 10 populations, 1 variable (classification). This variable gives the name of the continent in which a given population is located.
Source


Examples

data(humDNAm)
dpcoahum <- dpcoa(data.frame(t(humDNAm$samples)),
                  sqrt(humDNAm$distances), scan = FALSE, nf = 2)
plot(dpcoahum)

---

ichthyopoint sampling of fish community

Description

This data set gives informations between a faunistic array, the total number of sampling points made at each sampling occasion and the year of the sampling occasion.

Usage

data(ichthyo)

Format

*ichthyo* is a list of 3 components.

- **tab** is a faunistic array with 9 columns and 32 rows.
- **eff** is a vector of the 32 sampling effort.
- **dat** is a factor where the levels are the 10 years of the sampling occasion.

Details

The value \( n(i,j) \) at the \( i \)th row and the \( j \)th column in \( \text{tab} \) corresponds to the number of sampling points of the \( i \)th sampling occasion (in \( \text{eff} \)) that contains the \( j \)th species.

Source


Examples

data(ichthyo)
dudi1 <- dudi.dec(ichthyo$tab, ichthyo$eff, scannf = FALSE)
s.class(dudi1$li, ichthyo$dat, wt = ichthyo$eff / sum(ichthyo$eff))
Description

Computes the decomposition of inertia to measure the contributions of row and/or columns in multivariate methods.

Usage

```r
## S3 method for class 'dudi'
inertia(x, row.inertia = FALSE, col.inertia = FALSE, ...)
## S3 method for class 'inertia'
print(x, ...)
## S3 method for class 'inertia'
summary(object, sort.axis = 1, subset = 5, ...)
```

Arguments

- `x, object`: a duality diagram, object of class `dudi` for `inertia.dudi`. An object of class `inertia` for the methods `print` and `summary`
- `row.inertia`: if TRUE, returns the decomposition of inertia for the rows
- `col.inertia`: if TRUE, returns the decomposition of inertia for the columns
- `sort.axis`: the kept axis used to sort the contributions in decreasing order
- `subset`: the number of rows and/or columns to display in the summary
- `...`: further arguments passed to or from other methods

Details

Contributions are printed in percentage and the sign is the sign of the coordinates.

Value

An object of class `inertia`, i.e. a list containing:

- `tot.inertia`: repartition of the total inertia between axes
- `row.contrib`: contributions of the rows to the total inertia
- `row.abs`: absolute contributions of the rows (i.e. decomposition per axis)
- `row.rel`: relative contributions of the rows
- `row.cum`: cumulative relative contributions of the rows (i.e. decomposition per row)
- `col.contrib`: contributions of the columns to the total inertia
- `col.abs`: absolute contributions of the columns (i.e. decomposition per axis)
- `col.rel`: relative contributions of the columns
- `col.cum`: cumulative relative contributions of the columns (i.e. decomposition per column)
- `nf`: the number of kept axes
Author(s)

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(housetasks)
coa1 <- dudi.coa(housetasks, scann = FALSE)
res <- inertia(coa1, col = TRUE, row = FALSE)
res
summary(res)

data(irishdata)  

Geary’s Irish Data

Description

This data set contains geographical informations about 25 counties of Ireland.

Usage

data(irishdata)
irishdata

Format

irishdata is a list of 13 components:

area a data frame with polygons for each of the 25 contiguous counties
county.names a vector with the names of the 25 counties
xy a data frame with the coordinates centers of the 25 counties
tab a data frame with 25 rows (counties) and 12 variables
contour a data frame with the global polygon of all the 25 counties
link a matrix containing the common length between two counties from area
area.utm a data frame with polygons for each of the 25 contiguous counties expressed in Universal Transverse Mercator (UTM) coordinates
xy.utm a data frame with the UTM coordinates centers of the 25 counties
link.utm a matrix containing the common length between two counties from area.utm
tab.utm a data frame with the 25 counties (explicitly named) and 12 variables
contour.utm a data frame with the global polygon of all the 25 counties expressed in UTM coordinates
Spatial the map of the 25 counties of Ireland (an object of the class SpatialPolygons of sp)
Spatial.contour the contour of the map of the 25 counties of Ireland (an object of the class SpatialPolygons of sp)

Source


Examples

data(irishdata)

if(aderaphicsLoaded()) {

  if(requireNamespace("sp", quietly = TRUE)){
    g1 <- s.label(irishdata$xy.utm, Sp = irishdata$Spatial, pSp.col = "white", plot = FALSE)
    g21 <- s.label(irishdata$xy.utm, Sp = irishdata$Spatial, pSp.col = "white", plab.cex = 0,
                   ppoints.cex = 0, plot = FALSE)
    g22 <- s.label(irishdata$xy.utm, Sp = irishdata$Spatial.contour, pSp.col = "transparent",
                   plab.cex = 0, ppoints.cex = 0, pSp.lwd = 3, plot = FALSE)
    g2 <- superpose(g21, g22)
    g3 <- s.corcircle(dudi.pca(irishdata$tab, scan = FALSE)$co, plot = FALSE)
    score <- dudi.pca(irishdata$tab, scannf = FALSE, nf = 1)$li$Axis1
    names(score) <- row.names(irishdata$Spatial)
is.euclid

Is a Distance Matrix Euclidean?

Description

Confirmation of the Euclidean nature of a distance matrix by the Gower's theorem. is.euclid is used in summary.dist.

Usage

is.euclid(distmat, plot = FALSE, print = FALSE, tol = 1e-07)
## S3 method for class 'dist'
summary(object, ...)

Arguments

distmat an object of class 'dist'
plot a logical value indicating whether the eigenvalues bar plot of the matrix of the term \( -\frac{1}{2}d_{ij}^2 \) centred by rows and columns should be displayed
print a logical value indicating whether the eigenvalues of the matrix of the term \( -\frac{1}{2}d_{ij}^2 \) centred by rows and columns should be printed
tol a tolerance threshold: an eigenvalue is considered positive if it is larger than \(-tol*lambda1\) where lambda1 is the largest eigenvalue.
object an object of class 'dist'
... further arguments passed to or from other methods

Value

returns a logical value indicating if all the eigenvalues are positive or equal to zero
**Description**

This data set gives the spatial distribution of seeds (quadrats counts) of seven species in the understory of tropical rainforest.

**Usage**

data(julliot)

**Format**

julliot is a list with the following components:

- **tab** a data frame with 160 rows (quadrats) and 7 variables (species)
- **xy** a data frame with the coordinates of the 160 quadrats (positioned by their centers)
- **area** a data frame with 3 variables returning the boundary lines of each quadrat. The first variable is a factor. The levels of this one are the row.names of tab. The second and third variables return the coordinates (x,y) of the points of the boundary line.
- **Spatial** an object of the class SpatialPolygons of sp, containing the map

**Details**

Species names of julliot$tab are: *Pouteria torta, Minquartia guianensis, Quina obovata, Chrysophyllum lucentifolium, Parahancornia fasciculata, Virola michelii*, and *Pourouma spp.*
References


Examples

data(julliot)

```r
## Not run:
if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    obj1 <- sp::SpatialPolygonsDataFrame(Sr = julliot$Spatial, data = log(julliot$tab + 1))
    g1 <- s.Spatial(obj1)
    g2 <- s.value(julliot$xy, scalewt(log(julliot$tab + 1)), Sp = julliot$Spatial,
                  pSp.col = "white", pgrid.draw = FALSE)
  }
  if(requireNamespace("splancs", quietly = TRUE)) {
    par(mfrow = c(3, 3))
    for(k in 1:7)
      area.plot(julliot$area, val = log(julliot$tab[, k] + 1),
                 sub = names(julliot$tab)[k], csub = 2.5)
    par(mfrow = c(1, 1))
    for(k in 1:7) {
      area.plot(julliot$area)
      s.value(julliot$xy, scalewt(log(julliot$tab[, k] + 1)),
               sub = names(julliot$tab)[k], csub = 2.5, add.p = TRUE)
    }
    par(mfrow = c(1, 1))
  }
} else {
  if(requireNamespace("splancs", quietly = TRUE)) {
    par(mfrow = c(3, 3))
    for(k in 1:7)
      s.image(julliot$xy, log(julliot$tab[, k] + 1), kgrid = 3, span = 0.25, sub = names(julliot$tab)[k], csub = 2.5)
  }
} ## End(Not run)
```

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g3 <- s.image(julliot$xy, log(julliot$tab + 1), span = 0.25)
  }
  g4 <- s.value(julliot$xy, log(julliot$tab + 1))
} else {
  if(requireNamespace("splancs", quietly = TRUE)) {
    par(mfrow = c(3, 3))
    for(k in 1:7)
      s.image(julliot$xy, log(julliot$tab[, k] + 1), kgrid = 3, span = 0.25, sub = names(julliot$tab)[k], csub = 2.5)
  }
}
## K-tables Multi-Regions

### Description

This data set gives physical and physico-chemical variables, fish species, spatial coordinates about 92 sites.

### Usage

```r
data(jv73)
```

### Format

`jv73` is a list with the following components:

- **morpho**: a data frame with 92 sites and 6 physical variables
- **phychi**: a data frame with 92 sites and 12 physico-chemical variables
- **poi**: a data frame with 92 sites and 19 fish species
- **xy**: a data frame with 92 sites and 2 spatial coordinates
contour  a data frame for mapping  
fac.riv  a factor distributing the 92 sites on 12 rivers  
Spatial  an object of the class SpatialLines of sp, containing the map  

Source  

References  

Examples  

```r  
data(jv73)  
w <- split(jv73$morpho, jv73$fac.riv)  
w <- lapply(w, function(x) t(dudi.pca(x, scann = FALSE)))  
w <- ktab.list.dudi(w)  
if(adegraphicsLoaded()) {  
  if(requireNamespace("sp", quietly = TRUE)) {  
    g11 <- s.label(jv73$xy, Sp = jv73$Spatial, pori.incl = FALSE, plab.cex = 0.75, plot = FALSE)  
    g12 <- s.class(jv73$xy, jv73$fac.riv, ellipseSize = 0, pellipses.axes.draw = FALSE,  
                   starSize = 0, ppoints.cex = 0, plab.cex = 1.25, plot = FALSE)  
    g1 <- superpose(g11, g12, plot = TRUE)  
    g2 <- kplot(sepan(w), perm = TRUE, row.plab.cex = 0, posieig = "none")  
  }  
  }  
  s.label(jv73$xy, contour = jv73$contour, incl = FALSE, clab = 0.75)  
  s.class(jv73$xy, jv73$fac.riv, add.p = TRUE, cell = 0, axese = FALSE, csta = 0,  
           cpoi = 0, clab = 1.25)  
  kplot(sepan(w), perm = TRUE, clab.r = 0, clab.c = 2, show = FALSE)  
}  
```

The data frame `kcponds` contains informations about 33 ponds in De Maten reserve (Genk, Belgium).  

Description  

This data set contains informations about 33 ponds in De Maten reserve (Genk, Belgium).  

Usage  

```r  
data(kcponds)  
```
**Format**

kcponds is a list with the following components:

- **tab** a data frame with 15 environmental variables (columns) on 33 ponds (rows)
- **area** an object of class area
- **xy** a data frame with the coordinates of ponds
- **neig** an object of class neig
- **nb** the neighbourhood graph of the 33 sites (an object of class nb)
- **Spatial** an object of the class SpatialPolygons of sp, containing the map

**Details**

Variables of kcponds$tab are the following ones: depth, area, O2 (oxygen concentration), cond (conductivity), pH, Fe (Fe concentration), secchi (Secchi disk depth), N (NNO concentration), TP (total phosphorus concentration), chla (chlorophyll-a concentration), EM (emergent macrophyte cover), FM (floating macrophyte cover), SM (submerged macrophyte cover), denMI (total density of macroinvertebrates), divMI (diversity macroinvertebrates)

**Source**


**Examples**

data(kcponds)
w <- as.numeric(scalewt(kcponds$tab$N))

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g1 <- s.label(kcponds$xy, Sp = kcponds$Spatial, pSp.col = "white", nb = kcponds$nb,
                  plab.cex = 0, paxes.asp = "fill", plot = FALSE)
    g2 <- s.label(kcponds$xy, Sp = kcponds$Spatial, pSp.col = "white", plabels.cex = 0.8,
                  paxes.asp = "fill", plot = FALSE)
    g3 <- s.value(kcponds$xy, w, psub.text = "Nitrogen concentration", paxe.asp = "fill",
                  plot = FALSE)
    G <- rbindADEg(g1, g2, g3, plot = TRUE)
  }
} else {
  par(mfrow=c(3, 1))
  area.plot(kcponds$area)
  s.label(kcponds$xy, add.p = TRUE, cpoi = 2, clab = 0)
  s.label(kcponds$xy, add.p = TRUE, cpoi = 3, clab = 0)
  s.label(kcponds$xy, add.p = TRUE, cpoi = 0, clab = 0, neig = kcponds$neig, cneig = 1)
  area.plot(kcponds$area)
  s.label(kcponds$xy, add.p = TRUE, clab = 1.5)
  s.value(kcponds$xy, w, cleg = 2, sub = "Nitrogen concentration", csub = 4,
possub = "topright", include = FALSE)
par(mfrow = c(1, 1))
}

## Not run:
par(mfrow = c(3, 1))
pca1 <- dudi.pca(kcponds$tab, scan = FALSE, nf = 4)
if(requireNamespace("spdep", quietly = TRUE)) {
  multil <- multispati(pca1, spdep::nb2listw(neig2nb(kcponds$neig)), scannf = FALSE, nfposi = 2,
                       nfnega = 1)
  summary(multil)
} par(mfrow = c(1, 1))
## End(Not run)

---

### kdist

#### the class of objects 'kdist' (K distance matrices)

**Description**

An object of class kdist is a list of distance matrices observed on the same individuals.

**Usage**

```r
kdist(...) epsi = 1e-07, upper = FALSE)
```

**Arguments**

- `...`: a sequence of objects of the class kdist.
- `epsi`: a tolerance threshold to test if distances are Euclidean (Gower's theorem) using $\lambda_n / \lambda_1$ is larger than -epsi.
- `upper`: a logical value indicating whether the upper of a distance matrix is used (TRUE) or not (FALSE).

**Details**

The attributes of a 'kdist' object are:
- `names`: the names of the distances
- `size`: the number of points between distances are known
- `labels`: the labels of points
- `euclid`: a logical vector indicating whether each distance of the list is Euclidean or not.
- `call`: a call order
- `class`: object 'kdist'

**Value**

returns an object of class 'kdist' containing a list of semidefinite matrices.
Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

# starting from a list of matrices
data(yanomama)
lapply(yanomama,class)
k1 = kdist(yanomama)
print(k1)

# giving the correlations of Mantel's test
cor(as.data.frame(k1))
pairs(as.data.frame(k1))

# starting from a list of objects 'dist'
data(friday87)
fri.w <- ktab.data.frame(friday87$fau, friday87$fau.blo,
     tabnames = friday87$tab.names)
fri.kd = lapply(1:10, function(x) dist.binary(fri.w[[x]],2))
names(fri.kd) = friday87$tab.names
unlist(lapply(fri.kd,class)) # a list of distances
fri.kd = kdist(fri.kd)
fri.kd
s.corcircle(dudi.pca(as.data.frame(fri.kd), scan = FALSE)$co)

# starting from several distances
data(ecomor)
d1 <- dist.binary(ecomor$habitat, 1)
d2 <- dist.prop(ecomor$forsub, 5)
d3 <- dist.prop(ecomor$diet, 5)
d4 <- dist.quant(ecomor$morpho, 3)
d5 <- dist.taxo(ecomor$taxo)
ecomor.kd <- kdist(d1, d2, d3, d4, d5)
names(ecomor.kd) = c("habitat", "forsub", "diet", "morpho", "taxo")
class(ecomor.kd)
s.corcircle(dudi.pca(as.data.frame(ecomor.kd), scan = FALSE)$co)

data(bsetal97)
X <- prep.fuzzy.var(bsetal97$biol, bsetal97$biol.blo)
w1 <- attr(X, "col.num")
w2 <- levels(w1)
w3 <- lapply(w2, function(x) dist.quant(X[,w1==x], method = 1))
names(w3) <- names(attr(X, "col.blocks"))
w3 <- kdist(list = w3)
s.corcircle(dudi.pca(as.data.frame(w3), scan = FALSE)$co)
data(rpjdl)
w1 = lapply(1:10, function(x) dist.binary(rpjdl$fau, method = x))
w2 = c("JACCARD", "SOKAL_MICHENER", "SOKAL_SNEATH_S4", "ROGERS_TANIMOTO")
w2 = c(w2, "CZEKANOWSKI", "S9_GOWER_LEGENDRE", "OCHIAI", "SOKAL_SNEATH_S13")
w2 <- c(w2, "Phi_PEARSON", "S2_GOWER_LEGENDRE")
names(w1) <- w2
w3 = kdist(list = w1)
w4 <- dudi.pca(as.data.frame(w3), scan = FALSE)$co
w4

kdist2ktab

Transformation of K distance matrices (object 'kdist') into K Euclidean representations (object 'ktab')

Description

The function creates a ktab object with the Euclidean representations from a kdist object. Notice that the euclid attribute must be TRUE for all elements.

Usage

kdist2ktab(kd, scale = TRUE, tol = 1e-07)

Arguments

kd an object of class kdist
scale a logical value indicating whether the inertia of Euclidean representations are equal to 1 (TRUE) or not (FALSE).
tol a tolerance threshold, an eigenvalue is considered equal to zero if eig$values > (eig$values[1]*tol)

Value

returns a list of class ktab containing for each distance of kd the data frame of its Euclidean representation

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Examples

data(friday87)
fri.w <- ktab.data.frame(friday87$fau, friday87$fau.blo, tabnames = friday87$tab.names)
fri.kd <- lapply(1:10, function(x) dist.binary(fri.w[[x]], 10))
names(fri.kd) <- substr(friday87$tab.names, 1, 4)
fri.kd <- kdist(fri.kd)
fri.ktab <- kdist2ktab(kd = fri.kd)
fri.sepan <- sepan(fri.ktab)
plot(fri.sepan)

apply(fri.sepan$Eig, 1, sum)  # the sum of the eigenvalues is constant and equal to 1, for each K tables

fri.statis <- statis(fri.ktab, scan = FALSE, nf = 2)
round(fri.statis$RV, dig = 2)

fri.mfa <- mfa(fri.ktab, scan = FALSE, nf = 2)
fri.mcoa <- mcoa(fri.ktab, scan = FALSE, nf = 2)

apply(fri.statis$RV, 1, mean)
fri.statis$RV.tabw
plot(apply(fri.statis$RV, 1, mean), fri.statis$RV.tabw)
plot(fri.statis$RV.tabw, fri.statis$RV.tabw)

kdisteuclid

kdisteuclid  a way to obtain Euclidean distance matrices

Description

a way to obtain Euclidean distance matrices

Usage

kdisteuclid(obj, method = c("lingoes", "cailliez", "quasi"))

Arguments

obj  an object of class kdist
method a method to convert a distance matrix in a Euclidean one

Value

returns an object of class kdist with all distances Euclidean.

Author(s)

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
References


Examples

```r
w <- c(0.8, 0.8, 0.377350269, 0.8, 0.377350269, 0.377350269) # see ref.
w <- kdist(w)
w1 <- c(kdisteuclid(kdist(w), "lingoes"), kdisteuclid(kdist(w), "cailliez"),
       kdisteuclid(kdist(w), "quasi"))
print(w, print = TRUE)
print(w1, print = TRUE)

data(eurodist)
par(mfrow = c(1, 3))
eu1 <- kdist(eurodist) # an object of class 'dist'
plot(data.frame(unclass(c(eu1, kdisteuclid(eu1, "quasi")))), asp = 1)
title(main = "Quasi")
abline(0,1)
plot(data.frame(unclass(c(eu1, kdisteuclid(eu1, "lingoes")))), asp = 1)
title(main = "Lingoes")
abline(0,1)
plot(data.frame(unclass(c(eu1, kdisteuclid(eu1, "cailliez")))), asp = 1)
title(main = "Cailliez")
abline(0,1)
```

kplot

*Generic Function for Multiple Graphs in a K-tables Analysis*

Description

Methods for foucart, mcoa, mfa, pta, sepan, sepan.coa and statis

Usage

```r
kplot(object, ...)
```
Arguments

object an object used to select a method
... further arguments passed to or from other methods

Examples

methods(plot)
methods(scatter)
methods(kplot)

---

kplot.foucart Multiple Graphs for the Foucart’s Correspondence Analysis

Description

performs high level plots of a Foucart’s Correspondence Analysis, using an object of class foucart.

Usage

## S3 method for class 'foucart'
kplot(object, xax = 1, yax = 2, mfrow = NULL,
      which.tab = 1:length(object$blo), clab.r = 1, clab.c = 1.25,
      csub = 2, possub = "bottomright", ...)

Arguments

object an object of class foucart
xax, yax the numbers of the x-axis and the y-axis
mfrow a vector of the form 'c(nr,nc)', otherwise computed by as special own function
      n2mfrow
which.tab vector of table numbers for analyzing
clab.r a character size for the row labels
clab.c a character size for the column labels
csub a character size for the sub-titles used with par("cex")*csub
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
... further arguments passed to or from other methods
Examples

```r
data(bf88)
fou1 <- foucart(bf88, scann = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  g <- kplot(fou1, row.plab.cex = 0, psub.cex = 2)
} else {
  kplot(fou1, clab.c = 2, clab.r = 0, csub = 3)
}
```

kplot.mcoa  

**Multiple Graphs for a Multiple Co-inertia Analysis**

Description

performs high level plots of a Multiple Co-inertia Analysis, using an object of class `mcoa`.

Usage

```r
## S3 method for class 'mcoa'
kplot(object, xax = 1, yax = 2, which.tab = 1:nrow(object$cov2),
  mfrow = NULL, option = c("points", "axis", "columns"),
  clab = 1, cpoint = 2, csub = 2, possub = "bottomright", ...)
```

Arguments

- `object`: an object of class `mcoa`
- `xax, yax`: the numbers of the x-axis and the y-axis
- `which.tab`: a numeric vector containing the numbers of the tables to analyse
- `mfrow`: a vector of the form `c(nr, nc)`, otherwise computed by as special own function `n2mfrow`
- `option`: a string of characters for the drawing option
  - "points": plot of the projected scattergram onto the co-inertia axes
  - "axis": projections of inertia axes onto the co-inertia axes.
  - "columns": projections of variables onto the synthetic variables planes.
- `clab`: a character size for the labels
- `cpoint`: a character size for plotting the points, used with `par("cex") * cpoint`. If zero, no points are drawn.
- `csub`: a character size for the sub-titles, used with `par("cex") * csub`
- `possub`: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- `...`: further arguments passed to or from other methods
kplot.mfa

**Author(s)**
Daniel Chessel

**Examples**

```r
data(friday87)
w1 <- data.frame(scale(friday87$fau, scal = FALSE))
w2 <- ktab.data.frame(w1, friday87$fau.blo, tabnames = friday87$tab.names)
mcoa1 <- mcoa(w2, "lambda1", scan = FALSE)
kplot(mcoa1, option = "axis")
kplot(mcoa1)
kplot(mcoa1, option = "columns")
```

---

**kplot.mfa**

*Multiple Graphs for a Multiple Factorial Analysis*

**Description**

performs high level plots of a Multiple Factorial Analysis, using an object of class mfa.

**Usage**

```r
## S3 method for class 'mfa'
kplot(object, xax = 1, yax = 2, mfrow = NULL,
       which.tab = 1:length(object$blo), row.names = FALSE, col.names = TRUE,
       traject = FALSE, permute.row.col = FALSE,
       clab = 1, csub = 2, possub = "bottomright", ...)
```

**Arguments**

- `object`: an object of class mfa
- `xax`, `yax`: the numbers of the x-axis and the y-axis
- `mfrow`: a vector of the form `c(nr,nc)`, otherwise computed by a special own function `n2mfrow`
- `which.tab`: vector of the numbers of tables used for the analysis
- `row.names`: a logical value indicating whether the row labels should be inserted
- `col.names`: a logical value indicating whether the column labels should be inserted
- `traject`: a logical value indicating whether the trajectories of the rows should be drawn in a natural order
- `permute.row.col`: if TRUE, the rows are represented by vectors and columns by points, otherwise it is the opposite
- `clab`: a character size for the labels
- `csub`: a character size for the sub-titles, used with `par("cex")*csub`
- `possub`: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- `...`: further arguments passed to or from other methods
Author(s)

Daniel Chessel

Examples

data(friday87)
w1 <- data.frame(scale(friday87$fau, scal = FALSE))
w2 <- ktab.data.frame(w1, friday87$fau.blo, tabnames = friday87$tab.names)
mfa1 <- mfa(w2, scann = FALSE)
kplot(mfa1)

kplot.pta  Multiple Graphs for a Partial Triadic Analysis

Description

performs high level plots of a Partial Triadic Analysis, using an object of class pta.

Usage

## S3 method for class 'pta'
kplot(object, xax = 1, yax = 2, which.tab = 1:nrow(object$RV),
mfrow = NULL, which.graph = 1:4, clab = 1, cpoint = 2, csub = 2,
possub = "bottomright", ask = par("ask"), ...)

Arguments

object an object of class pta
xax, yax the numbers of the x-axis and the y-axis
which.tab a numeric vector containing the numbers of the tables to analyse
mfrow parameter of the array of figures to be drawn, otherwise the graphs associated to a table are drawn on the same row
which.graph an option for drawing, an integer between 1 and 4. For each table of which.tab, are drawn:
  1 the projections of the principal axes
  2 the projections of the rows
  3 the projections of the columns
  4 the projections of the principal components onto the planes of the compromise
clab a character size for the labels
cpoint a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn.
csub a character size for the sub-titles, used with par("cex")*csub
possuab a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
ask a logical value indicating if the graphs requires several arrays of figures
... further arguments passed to or from other methods
Author(s)

Daniel Chessel

Examples

data(meaudret)
wit1 <- wca(dudi.pca(meaudret$spe, scan = FALSE, scal = FALSE),
            meaudret$design$season, scan = FALSE)
pta1 <- kta1 <- ktab.within(wit1, colnames = rep(c("S1", "S2", "S3", "S4", "S5"), 4))
pta2 <- t(kta1)
pta1 <- pta(pta2, scann = FALSE)
kplot(pta1)
kplot(pta1, which.graph = 3)

---

**Description**

performs high level plots for Separated Analyses in a K-tables, using an object of class `sepan`.

Usage

```r
## S3 method for class 'sepan'
kplot(object, xax = 1, yax = 2, which.tab = 1:length(object$blo),
       mfrow = NULL, permute.row.col = FALSE, clab.row = 1,
       clab.col = 1.25, traject.row = FALSE, csub = 2,
       possub = "bottomright", show.eigen.value = TRUE,...)

kplotsepan.coa(object, xax = 1, yax = 2, which.tab = 1:length(object$blo),
                mfrow = NULL, permute.row.col = FALSE, clab.row = 1,
                clab.col = 1.25, csub = 2, possub = "bottomright",
                show.eigen.value = TRUE, poseig = c("bottom", "top"), ...)
```

Arguments

- `object` an object of class `sepan`
- `xax, yax` the numbers of the x-axis and the y-axis
- `which.tab` a numeric vector containing the numbers of the tables to analyse
- `mfrow` parameter for the array of figures to be drawn, otherwise use `n2mfrow`
- `permute.row.col` if TRUE the rows are represented by arrows and the columns by points, if FALSE it is the opposite
- `clab.row` a character size for the row labels
- `clab.col` a character size for the column labels
traject.row a logical value indicating whether the trajectories between rows should be drawn in a natural order

csub a character size for the sub-titles, used with par("cex")*csub

possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

show.eigen.value a logical value indicating whether the eigenvalues bar plot should be drawn

poseig if "top" the eigenvalues bar plot is upside, if "bottom", it is downside

... further arguments passed to or from other methods

Details

kplot.sepan superimposes the points for the rows and the arrows for the columns using an adapted rescaling such as the scatter.dudi.

kplotsepan.coa superimposes the row coordinates and the column coordinates with the same scale.

Author(s)

Daniel Chessel

Examples

data(escopage)
  w1 <- data.frame(scale(escopage$tab))
  w1 <- ktab.data.frame(w1, escopage$blo, tabnames = escopage$tab.names)
  sep1 <- sepan(w1)
  if(adegraphicsLoaded()) {
    kplot(sep1, posieig = "none")
  } else {
    kplot(sep1, show = FALSE)
  }

data(friday87)
  w2 <- data.frame(scale(friday87$fau, scal = FALSE))
  w2 <- ktab.data.frame(w2, friday87$fau.blo, tabnames = friday87$tab.names)
  if(adegraphicsLoaded()) {
    kplot(sepan(w2), row.plabel.cex = 1.25, col.plab.cex = 0)
  } else {
    kplot(sepan(w2), clab.r = 1.25, clab.c = 0)
  }

data(microsatt)
  w3 <- dudi.coa(data.frame(t(microsatt$tab)), scann = FALSE)
  loci.fac <- factor(rep(microsatt$loci.names, microsatt$loci.eff))
  wit <- wca(w3, loci.fac, scann = FALSE)
  microsatt.ktab <- ktab.within(wit)
  if(adegraphicsLoaded()) {
    kplotsepan.coa(sepan(microsatt.ktab), posieig = "none", col.plab.cex = 0, row.plab.cex = 1.5)
  } else {
kplot.statis

kplot(coa(sepan(microsatt.ktab), show = FALSE, clab.c = 0, mfrow = c(3,3), clab.r = 1.5))

---

dkplot.statis

Multiple Graphs of a STATIS Analysis

Description

performs high level plots for a STATIS analysis, using an object of class `statis`.

Usage

```r
## S3 method for class 'statis'
kplot(object, xax = 1, yax = 2, mfrow = NULL, which.tab = 1:length(object$tab.names), clab = 1.5, cpoi = 2, traject = FALSE, arrow = TRUE, class = NULL, unique.scale = FALSE, csub = 2, possub = "bottomright", ...)
```

Arguments

- **object**: an object of class `statis`
- **xax**, **yax**: the numbers of the x-axis and the y-axis
- **mfrow**: parameter for the array of figures to be drawn
- **which.tab**: a numeric vector containing the numbers of the tables to analyse
- **clab**: a character size for the labels
- **cpoi**: the size of points
- **traject**: a logical value indicating whether the trajectories should be drawn in a natural order
- **arrow**: a logical value indicating whether the column factorial diagrams should be plotted
- **class**: if not NULL, a factor of length equal to the number of the total columns of the K-tables
- **unique.scale**: if TRUE, all the arrays of figures have the same scale
- **csub**: a character size for the labels of the arrays of figures used with `par("cex")*csub`
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- **...**: further arguments passed to or from other methods

Author(s)

Daniel Chessel
Examples

```r
data(jv73)
dudi1 <- dudi.pca(jv73$poi, scann = FALSE, scal = FALSE)
wit1 <- wca(dudi1, jv73$fac.riv, scann = FALSE)
ktal1 <- ktab.within(wit1)
statis1 <- statis(ktal1, scann = FALSE)
if(adegraphicsLoaded()) {
g1 <- kplot(statis1, traj = TRUE, arrow = FALSE, plab.cex = 0, psub.cex = 2, ppoi.cex = 2)
} else {
kplot(statis1, traj = TRUE, arrow = FALSE, unique = TRUE, clab = 0, csub = 2, cpoi = 2)
}
```

krandtest

Class of the Permutation Tests (in C).

Description

Plot, print and extract permutation tests. Objects of class 'krandtest' are lists.

Usage

```r
as.krandtest(sim, obs, alter = "greater", call = match.call(),
             names = colnames(sim), p.adjust.method = "none", output = c("light", "full"))
```

Arguments

- **sim**: a matrix or data.frame of simulated values (repetitions as rows, number of tests as columns
- **obs**: a numeric vector of observed values for each test
- **alter**: a vector of character specifying the alternative hypothesis for each test. Each element must be one of "greater" (default), "less" or "two-sided". The length must be equal to the length of the vector obs, values are recycled if shorter.
- **call**: a call order
- **names**: a vector of names for tests
- **p.adjust.method**: a string indicating a method for multiple adjustment, see `p.adjust.methods` for possible choices.
output a character string specifying if all simulations should be stored ("full"). This was the default until ade4 1.7-5. Now, by default ("light"), only the distribution of simulated values is stored in element plot as produced by the hist function.

x an object of class 'krandtest'
mfrow a vector of the form 'c(nr,nc)', otherwise computed by as special own function
nclass a number of intervals for the histogram. Ignored if object output is "light"
main.title a string of character for the main title
... further arguments passed to or from other methods
i numeric indices specifying elements to extract

Value

plot.krandtest draws the $p$ simulated values histograms and the position of the observed value.

[.].krandtest returns a krandtest object and [[.] krandtest returns a randtest object.

Author(s)
Daniel Chessel and Stéphane Dray <stephane.dray@univ-lyon1.fr>

See Also
randtest

Examples

wkrandtest <- as.krandtest(obs = c(0, 1.2, 2.4, 3.4, 5.4, 20.4),
    sim = matrix(rnorm(6*200), 200, 6))
wkrandtest
plot(wkrandtest)
wkrandtest[[c(1, 4, 6)]]
wkrandtest[[[1]]]

ktab the class of objects 'ktab' (K-tables)

Description

an object of class ktab is a list of data frames with the same row.names in common.
a list of class 'ktab' contains moreover:

blo : the vector of the numbers of columns for each table
lw : the vector of the row weightings in common for all tables
cw : the vector of the column weightings
**TL**: a data frame of two components to manage the parameter positions associated with the rows of tables

**TC**: a data frame of two components to manage the parameter positions associated with the columns of tables

**T4**: a data frame of two components to manage the parameter positions of 4 components associated to an array

### Usage

```r
## S3 method for class 'ktab'
c(...)
## S3 method for class 'ktab'
x[i,j,k]
is.ktab(x)
## S3 method for class 'ktab'
t(x)
## S3 method for class 'ktab'
row.names(x)
## S3 method for class 'ktab'
col.names(x)
tab.names(x)
col.names(x)
tab.util.names(x)
```

### Arguments

- **x**: an object of the class ktab
- **...**: a sequence of objects of the class ktab
- **1,j,k**: elements to extract (integer or empty): index of tables (i), rows (j) and columns (k)

### Details

A 'ktab' object can be created with:
- a list of data frame: `ktab.list.df`
- a list of dudi objects: `ktab.list.dudi`
- a data.frame: `ktab.data.frame`
- an object within: `ktab.within`
- a couple of ktabs: `ktab.match2ktabs`

### Value

- `c.ktab` returns an object ktab. It concatenates K-tables with the same rows in common.
- `t.ktab` returns an object ktab. It permutes each data frame into a K-tables. All tables have the same column names and the same column weightings (a data cube).
- `"[" returns an object ktab. It allows to select some arrays in a K-tables.
- `is.ktab` returns TRUE if x is a K-tables.
row.names returns the vector of the row names common with all the tables of a K-tables and allows to modifie them.
col.names returns the vector of the column names of a K-tables and allowes to modifie them.
tab.names returns the vector of the array names of a K-tables and allows to modifie them.
ktab.util.names is a useful function.

Author(s)
Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr> Stéphane Dray <stephane.dray@univ-lyon1.fr>

Examples

data(friday87)
wfri <- data.frame(scale(friday87$fau, scal = FALSE))
wfri <- ktab.data.frame(wfri, friday87$fau.blo)
wfri[2:4, 1:5, 1:3]
c(wfri[2:4], wfri[5])

data(meaudret)
wit1 <- withinpca(meaudret$env, meaudret$design$season, scan = FALSE, 
scal = "partial")
kta1 <- ktab.within(wit1, colnames = rep(c("S1", "S2", "S3", "S4", "S5"), 4))
kta2 <- t(kta1)

if(adegraphicsLoaded()){
  kplot(sepan(kta2), row.plab.cex = 1.5, col.plab.cex = 0.75)
} else {
  kplot(sepan(kta2), clab.r = 1.5, clab.c = 0.75)
}

ktab.data.frame

Creation of K-tables from a data frame

description
creates K tables from a data frame.

Usage

ktab.data.frame(df, blocks, rownames = NULL, colnames = NULL, 
tabnames = NULL, w.row = rep(1, nrow(df)) / nrow(df),
    w.col = rep(1, ncol(df)))

Arguments

df a data frame
blocks an integer vector for which the sum must be the number of variables of df. Its
length is the number of arrays of the K-tables
ktab.list.df

Creating a K-tables from a list of data frames.

Description

creates a list of class ktab from a list of data frames

Usage

```r
ktab.list.df(obj, rownames = NULL, colnames = NULL, tabnames = NULL,
             w.row = rep(1, nrow(obj[[1]])), w.col = lapply(obj, function(x)
             rep(1 / ncol(x), ncol(x))))
```

Arguments

- **obj**: a list of data frame
- **rownames**: the names of the K-tables rows (otherwise, the row names of the arrays)
- **colnames**: the names of the K-tables columns (otherwise, the column names of the arrays)
- **tabnames**: the names of the arrays of the K-tables (otherwise, the names of the obj if they exist, or else "Ana1", "Ana2", . . .)
- **w.row**: a vector of the row weightings in common with all the arrays
- **w.col**: a list of the vector of the column weightings for each array
Details

Each element of the initial list have to possess the same names and row numbers.

Value

returns a list of class ktab. See ktab

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

Examples

data(jv73)
l0 <- split(jv73$morpho, jv73$fac.riv)
l0 <- lapply(l0, function(x) data.frame(t(scalewt(x))))
kta <- ktab.list.df(l0)
kplot(sepan(kta[c(2, 5, 7, 10)]), perm = TRUE)

Description

creates a list of class ktab from a list of duality diagrams.

Usage

ktab.list.dudi(obj, rownames = NULL, colnames = NULL, tabnames = NULL)

Arguments

obj a list of objects of class 'dudi'. Each element of the list must have the same row names for $tab and even for $lw
rownames the row names of the K-tables (otherwise the row names of the $tab)
colnames the column names of the K-tables (otherwise the column names of the $tab)
tabnames the names of the arrays of the K-tables (otherwise the names of the obj if they exist, or else "Ana1", "Ana2", ...)

Value

returns a list of class ktab. See ktab

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Examples

data(euro123)
 pca1 <- dudi.pca(euro123$in78, scale = FALSE, scann = FALSE)
 pca2 <- dudi.pca(euro123$in86, scale = FALSE, scann = FALSE)
 pca3 <- dudi.pca(euro123$in97, scale = FALSE, scann = FALSE)
 ktabeuro <- ktab.list.dudi(list(pca1, pca2, pca3),
     tabnames = c("1978", "1986", "1997"))
 if(adegraphicsLoaded()) {
   kplot(sepan(ktabeuro))
 } else {
   kplot(sepan(ktabeuro), mfr = c(2, 2), clab.c = 1.5)
 }

data(meaudret)
w1 <- split(meaudret$env, meaudret$design$season)
 ll <- lapply(w1, dudi.pca, scann = FALSE)
 kta <- ktab.list.dudi(ll, rownames <- paste("Site", 1:5, sep = ""))
 if(adegraphicsLoaded()) {
   kplot(sepan(kta), row.plab.cex = 1.5, col.plab.cex = 0.75)
 } else {
   kplot(sepan(kta), clab.r = 1.5, clab.c = 0.75)
 }

data(jv73)
w <- split(jv73$poi, jv73$fac.riv)
wjv73poi <- lapply(w, dudi.pca, scal = FALSE, scan = FALSE)
wjv73poi <- lapply(wjv73poi, t)
wjv73poi <- ktab.list.dudi(wjv73poi)
kplot(sepan(wjv73poi), permut = TRUE, traj = TRUE)

ktab.match2ktabs

STATIS and Co-Inertia: Analysis of a series of paired ecological tables

Description

Prepares the analysis of a series of paired ecological tables. Partial Triadic Analysis (see pta) can be used thereafter to perform the analysis of this k-table.

Usage

ktab.match2ktabs(KTX, KTY)

Arguments

KTX an objet of class ktab
KTY an objet of class ktab
ktab.within

**Value**

a list of class ktab, subclass kcoinertia. See ktab

**WARNING**

IMPORTANT : KTX and KTY must have the same k-tables structure, the same number of columns, and the same column weights.

**Author(s)**

Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

**References**


**Examples**

data(meau)
wit1 <- withinpca(meau$env, meau$design$season, scan = FALSE, scal = "total")
pcaspe <- dudi.pca(meau$spe, scale = FALSE, scan = FALSE, nf = 2)
wit2 <- wca(pcaspe, meau$design$season, scan = FALSE, nf = 2)
kta1 <- ktab.within(wit1, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kta2 <- ktab.within(wit2, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kcoi <- ktab.match2ktabs(kta1, kta2)
ptacoi <- pta(kcoi, scan = FALSE, nf = 2)
plot(ptacoi)
kplot(ptacoi)

---

**ktab.within**

*Process to go from a Within Analysis to a K-tables*

**Description**

performs the process to go from a Within Analysis to a K-tables.

**Usage**

ktab.within(dudiwit, rownames = NULL, colnames = NULL, tabnames = NULL)
Arguments

- dudiwit: an objet of class within
- rownames: the row names of the K-tables (otherwise the row names of dudiwit$tab)
- colnames: the column names of the K-tables (otherwise the column names of dudiwit$tab)
- tabnames: the names of the arrays of the K-tables (otherwise the levels of the factor which defines the within-classes)

Value

A list of class ktab. See ktab.

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

Examples

```r
data(bacteria)
w1 <- data.frame(t(bacteria$espcodon))
dudi1 <- dudi.coa(w1, scann = FALSE, nf = 4)
wit1 <- wca(dudi1, bacteria$code, scannf = FALSE)
ta1 <- ktab.within(wit1)
plot(statis(ta1, scann = FALSE))

kta2 <- ta1[ta1$blo>3]
kplot(mfa(kta2, scann = FALSE))
```

Description

This data set gives meristic, genetic and morphological data frame for 306 trouts.

Usage

data(lascaux)

Format

lascaux is a list of 9 components.

- riv: a factor returning the river where 306 trouts are captured
- code: vector of characters: code of the 306 trouts
- sex: factor sex of the 306 trouts
meris  data frame 306 trouts - 5 meristic variables

tap   data frame of the total number of red and black points

gen   factor of the genetic code of the 306 trouts

morpho data frame 306 trouts 37 morphological variables

colo  data frame 306 trouts 15 variables of coloring

ornem data frame 306 trouts 15 factors (ornamentation)

Source


References


Examples

data(lascaux)

if(adegraphicsLoaded()) {
  g1 <- s1d.barchart(dudi.pca(lascaux$meris, scan = FALSE)$eig, psub.text = "Meristic",
                     p1d.horizontal = FALSE, plot = FALSE)
  g2 <- s1d.barchart(dudi.pca(lascaux$colo, scan = FALSE)$eig, psub.text = "Coloration",
                     p1d.horizontal = FALSE, plot = FALSE)
  g3 <- s1d.barchart(dudi.pca(na.omit(lascaux$morpho), scan = FALSE)$eig, psub.text = "Morphometric",
                     p1d.horizontal = FALSE, plot = FALSE)
  g4 <- s1d.barchart(dudi.acm(na.omit(lascaux$orne), scan = FALSE)$eig, psub.text = "Ornemental",
                     p1d.horizontal = FALSE, plot = FALSE)
  G <- ADEeG5(c(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2,2))
  barplot(dudi.pca(lascaux$meris, scan = FALSE)$eig)
  title(main = "Meristic")
  barplot(dudi.pca(lascaux$colo, scan = FALSE)$eig)
  title(main = "Coloration")
  barplot(dudi.pca(na.omit(lascaux$morpho), scan = FALSE)$eig)
  title(main = "Morphometric")
  barplot(dudi.acm(na.omit(lascaux$orne), scan = FALSE)$eig)
  title(main = "Ornemental")
  par(mfrow = c(1,1))
}
**lingoes**  
*Transformation of a Distance Matrix for becoming Euclidean*

**Description**

transforms a distance matrix in a Euclidean one.

**Usage**

```r
lingoes(distmat, print = FALSE, tol = 1e-07, cor.zero = TRUE)
```

**Arguments**

- `distmat`: an object of class `dist`
- `print`: if TRUE, prints the eigenvalues of the matrix
- `tol`: a tolerance threshold for zero
- `cor.zero`: if TRUE, zero distances are not modified

**Details**

The function uses the smaller positive constant k which transforms the matrix of $\sqrt{d_{ij}^2 + 2 \times k}$ in an Euclidean one.

**Value**

returns an object of class `dist` with a Euclidean distance

**Author(s)**

Daniel Chessel  
Stéphane Dray <stephane.dray@univ-lyon1.fr>

**References**


**Examples**

```r
data(capitales)  
d0 <- capitales$dist  
is.euclid(d0) # FALSE  
d1 <- lingoes(d0, TRUE)  
# Lingoes constant = 2120982  
is.euclid(d1) # TRUE  
plot(d0, d1)  
x0 <- sort(unclass(d0))  
lines(x0, sqrt(x0^2 + 2 * 2120982), lwd = 3)
```
lizards

Phylogeny and quantitative traits of lizards

Description

This data set describes the phylogeny of 18 lizards as reported by Bauwens and Díaz-Uriarte (1997). It also gives life-history traits corresponding to these 18 species.

Usage

data(lizards)

Format

lizards is a list containing the 3 following objects:

- **traits** is a data frame with 18 species and 8 traits.
- **hprA** is a character string giving the phylogenetic tree (hypothesized phylogenetic relationships based on immunological distances) in Newick format.
- **hprB** is a character string giving the phylogenetic tree (hypothesized phylogenetic relationships based on morphological characteristics) in Newick format.

Details

Variables of lizards$traits are the following ones: mean.L (mean length (mm)), matur.L (length at maturity (mm)), max.L (maximum length (mm)), hatch.L (hatchling length (mm)), hatch.m (hatchling mass (g)), clutch.S (Clutch size), age.mat (age at maturity (number of months of activity)), clutch.F (clutch frequency).

References


Examples

```r
data(lizards)
w <- data.frame(scalewt(log(lizards$traits)))
par(mfrow = c(1,2))
wphy <- newick2phylog(lizards$hprA)
table.phylog(w, wphy, csi = 3)
wphy <- newick2phylog(lizards$hprB)
table.phylog(w, wphy, csi = 3)
par(mfrow = c(1,1))
```

---

`loocv.between`  
*Leave-one-out cross-validation for a bca*

Description

Leave-one-out cross-validation for `bca`.

Usage

```r
## S3 method for class 'between'
loocv(x, nax = 0, progress = FALSE, parallel = FALSE, ...)
```

Arguments

- `x`  
  dudi of the `bca` on which cross-validation should be done
- `nax`  
  list of axes for mean overlap index computation (0 = all axes)
- `progress`  
  logical, `TRUE` = display a progress bar during computations
- `parallel`  
  logical, `TRUE` = process cross-validation in parallel computing
- `...`  
  further arguments passed to or from other methods

Details

This function returns a list containing the cross-validated coordinates of the rows (the rows of the original analysis, not the rows of the `bca`). The `dudi` on which the `bca` was computed is redone after removing each row of the data table, one at a time. A `bca` is done on this new `dudi` and the coordinates of the missing row are computed by projection as supplementary element in the corresponding `bca`. This is most useful in the case `p > n` (many variables and few samples), where `bca` graphs can show spurious groups (see Refs.)

For parallel computing (`parallel` argument = `TRUE`), the new `dudi`, `bca` and cross-validation computations are processed in parallel on all the available nodes of the computer processor(s).
**Value**

A list with:
- `XValCoord`: the cross-validated row coordinates
- `PRESS`: the Predicted Residual Error Sum for each row
- `PRESSTot`: the sum of `PRESS` for each `bca` axis
- `Oij_bga`: the mean overlap index for BGA
- `Oij_XVal`: the mean overlap index for cross-validation
- `DeltaOij`: the spuriousness index

**Author(s)**

Jean Thioulouse

**References**


**See Also**

`loocv.dudi` `loocv.discrimin`

**Examples**

```r
# Data = meaudret
data(meaudret)
pca1 <- dudi.pca(meaudret$env, scannf = FALSE, nf = 3)
bca1 <- bca(pca1, meaudret$design$site, scannf = FALSE, nf = 3)
pst1 <- paste0("Meaudret BGA randtest: p=",
                randtest(bca1)$pvalue, " ratio=", round(bca1$ratio, 2))
xbca1 <- loocv(bca1, progress = TRUE)

if(adegraphicsLoaded()){
  sc1 <- s.class(bca1$ls, meaudret$design$site, col = TRUE, psub.text = pst1, ellipseSize=0, chullSize=1, plot = FALSE)
  sc2 <- s.class(xbca1$XValCoord, meaudret$design$site, col = TRUE, psub.text = "Meaudret cross-validation", ellipseSize=0, chullSize=1, plot = FALSE)
  ADEgS(list(sc1, sc2))
} else {
  par(mfrow=c(2,2))
s.chull(dfxy = bca1$ls, fac = meaudret$design$site, cpoint = 1, col = hcl.colors(5, "Dark 2"), sub = pst1)
  s.class(bca1$ls, meaudret$design$site, col = hcl.colors(5, "Dark 2"),
```

```r
classification = loocv.betavcoord(bca1, meaudret$design$site, progress = TRUE)
classification$Coord[classification$Coord$classification == "bga",]
classification$Coord[classification$Coord$classification == "xval",]
```
loocv.discrimin

Leave-one-out cross-validation for a discrimin analysis

Description

Leave-one-out cross-validation to test the existence of groups in a discrimin analysis.

Usage

## S3 method for class 'discrimin'
loocv(x, nax = 0, progress = FALSE, ...)
Arguments

- `x` the `discrimin` analysis on which cross-validation should be done
- `nax` list of axes for mean overlap index computation (0 = all axes)
- `progress` logical to display a progress bar during computations (see the `progress` package)
- `...` further arguments passed to or from other methods

Details

This function returns a list containing the cross-validated coordinates of the rows. The analysis on which the `discrimin` was computed is redone after removing each row of the data table, one at a time. A `discrimin` analysis is done on this new analysis and the coordinates of the missing row are computed by projection as supplementary element in the new `discrimin` analysis. This can be useful to check that the groups evidenced by the `discrimin` analysis are supported.

Value

A list with:
- `XValCoord`: the cross-validated row coordinates
- `PRESS`: the Predicted Residual Error Sum for each row
- `PRESSTot`: the sum of `PRESS` for each `bca` axis
- `Oij_disc`: the mean overlap index for the discriminant analysis
- `Oij_XVal`: the mean overlap index for cross-validation
- `DeltaOij`: the spuriousness index

Author(s)

Jean Thioulouse

See Also

- `loocv.dudi`
- `loocv.between`

Examples

```r
## Not run:
# Data = skulls
data(skulls)
pcaskul <- dudi.pca(skulls, scan = FALSE)
facskul <- gl(5,30)
diskul <- discrimin(pcaskul, facskul, scan = FALSE)
xdiskul <- loocv(diskul, progress = TRUE)
oijdisc <- xdiskul$Oij_disc
oijxval <- xdiskul$Oij_XVal
Doij <- (oijxval - oijdisc)/0.5*100
pst1 <- paste0("Skulls discrimin randtest: p=", round(randtest(diskul)$pvalue, 4), ", Oij = ", round(oijdisc,2))
```
loocv.dudi

Leave-one-out cross-validation for a dudi
Description

Leave-one-out cross-validation to check the dispersion of row coordinates in a dudi.

Usage

```r
## S3 method for class 'dudi'
loocv(x, progress = FALSE, ...)
```

Arguments

- `x` : the dudi of the bca on which cross-validation should be done
- `progress` : logical to display a progress bar during computations (see the `progress` package)
- `...` : further arguments passed to or from other methods

Details

This function does a cross-validation of the row coordinates of a dudi. Each row is removed from the table one at a time, and its coordinates are computed by projection of this row in the analysis of the table with the removed row. This can be used to check the sensitivity of an analysis to outliers. The cross-validated and original coordinates can be compared with the `s.match` function (see example).

Value

A list with:

- `XValCoord` : the cross-validated row coordinates
- `PRESS` : the Predicted Residual Error Sum for each row
- `PRESSTot` : the sum of PRESS for each bca axis

Author(s)

Jean Thioulouse

See Also

`loocv.between`, `loocv.discrimin`, `suprow`, `s.match`

Examples

```r
data(meaudret)
envpca <- dudi.pca(meaudret$env, scannf = FALSE, nf = 3)
xvpca <- loocv(envpca)
s.match(envpca$li, xvpca$XValCoord)
```
This data set gives the landmarks of a macaca at the ages of 0.9 and 5.77 years.

Usage

data(macaca)

Format

macaca is a list of 2 components.

xy1 is a data frame with 72 points and 2 coordinates.

xy2 is a data frame with 72 points and 2 coordinates.

Source


Examples

data(macaca)
pro1 <- procuste(macaca$xy1, macaca$xy2, scal = FALSE)
pro2 <- procuste(macaca$xy1, macaca$xy2)

if(adeGraphicsLoaded()) {
  g1 <- s.match(macaca$xy1, macaca$xy2, plab.cex = 0, plot = FALSE)
g2 <- s.match(pro1$tabX, pro1$rotY, plab.cex = 0.7, plot = FALSE)
g3 <- s.match(pro1$tabY, pro1$rotX, plab.cex = 0.7, plot = FALSE)
g4 <- s.match(pro2$tabY, pro2$rotX, plab.cex = 0.7, plot = FALSE)
G <- ADEgS(c(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2,2))
s.match(macaca$xy1, macaca$xy2, clab = 0)
s.match(pro1$tabX, pro1$rotY, clab = 0.7)
s.match(pro1$tabY, pro1$rotX, clab = 0.7)
s.match(pro2$tabY, pro2$rotX, clab = 0.7)
par(mfrow = c(1,1))
}
macon

Wine Tasting

Description

The macon data frame has 8 rows-wines and 25 columns-tasters. Each column is a classification of 8 wines (Beaujolais, France).

Usage

data(macon)

Source

Foire Nationale des Vins de France, Mâcon, 1985

Examples

data(macon)
s.corcircle(dudi.pca(macon, scan = FALSE)$co)

macroloire

Assemblages of Macroinvertebrates in the Loire River (France)

Description

A total of 38 sites were surveyed along 800 km of the Loire River yielding 40 species of Trichoptera and Coleoptera sampled from riffle habitats. The river was divided into three regions according to geology: granitic highlands (Region#1), limestone lowlands (Region#2) and granitic lowlands (Region#3). This data set has been collected for analyzing changes in macroinvertebrate assemblages along the course of a large river. Four criterias are given here: variation in 1/ species composition and relative abundance, 2/ taxonomic composition, 3/ Body Sizes, 4/ Feeding habits.

Usage

data(macroloire)

Format

macroloire is a list of 5 components.

fau is a data frame containing the abundance of each species in each station.
traits is a data frame describes two traits: the maximal sizes and feeding habits for each species. Each trait is divided into categories. The maximal size achieved by the species is divided into four length categories: \( \leq 5\text{mm} \); \( >5-10\text{mm} \); \( >10-20\text{mm} \); \( >20-40\text{mm} \). Feeding habits comprise seven categories: engulfers, shredders, scrapers, deposit-feeders, active filter-feeders, passive filter-feeders and piercers, in this order. The affinity of each species to each trait category is quantified using a fuzzy coding approach. A score is assigned to each species for describing its affinity for a given trait category from "0" which indicates no affinity to "3" which indicates high affinity. These affinities are further transformed into percentage per trait per species.

taxo is a data frame with species and 3 factors: Genus, Family and Order. It is a data frame of class "taxo": the variables are factors giving nested classifications.

envir is a data frame giving for each station, its name (variable "SamplingSite"), its distance from the source (km, variable "Distance"), its altitude (m, variable "Altitude"), its position regarding the dams [1: before the first dam; 2: after the first dam; 3: after the second dam] (variable "Dam"), its position in one of the three regions defined according to geology: granitic highlands, limestone lowlands and granitic lowlands (variable "Morphoregion"), presence of confluence (variable "Confluence")

labels is a data frame containing the latin names of the species.

Source


Examples

data(macroloire)
apqe.Equi <- apqe(macroloire$fau, , macroloire$morphoregions)
apqe.Equi
#test.Equi <- randtest.apqe(apqe.Equi, method = "aggregated", 99)
#plot(test.Equi)

## Not run:
m.phy <- taxo2phylog(macroloire$taxo)
apqe.Tax <- apqe(macroloire$fau, m.phy$Wdist, macroloire$morphoregions)
apqe.Tax
#test.Tax <- randtest.apqe(apqe.Tax, method = "aggregated", 99)
#plot(test.Tax)

dSize <- sqrt(dist.prop(macroloire$traits[,1:4], method = 2))
apqe.Size <- apqe(macroloire$fau, dSize, macroloire$morphoregions)
apqe.Size
#test.Size <- randtest.apqe(apqe.Size, method = "aggregated", 99)
#plot(test.Size)
dFeed <- sqrt(dist.prop(macroloire$traits[,-(1:4)], method = 2))
apqe.Feed <- apqe(macroloire$fau, dFeed, macroloire$morphoregions)
apqe.Feed
#test.Feed <- randtest.apqe(apqe.Feed, method = "aggregated", 99)
#plot(test.Size)

## End(Not run)

mafragh  

Phyto-Ecological Survey

Description

This data set gives environmental and spatial informations about species and sites.

Usage

data(mafragh)

Format

mafragh is a list with the following components:

- **xy** the coordinates of 97 sites
- **flo** a data frame with 97 sites and 56 species
- **neig** the neighbourhood graph of the 97 sites (an object of class `neig`)
- **env** a data frame with 97 sites and 11 environmental variables
- **partition** a factor classifying the 97 sites in 7 classes
- **area** a data frame of class `area`
- **tre** a character providing the phylogeny as a newick object
- **traits** a list of data frame. Each data frame provides the value of biological traits for plant species
- **nb** the neighbourhood graph of the 97 Mafragh sites (an object of class `nb`)
- **Spatial** the map of the 97 Mafragh sites (an object of the class `SpatialPolygons` of sp)
- **spenames** a data frame with 56 rows (species) and 2 columns (names)
- **Spatial.contour** the contour of the Magragh map (an object of the class `SpatialPolygons` of sp)

Source


References


Examples

data(mafragh)
coa1 <- dudi.coa(mafragh$flo, scan = FALSE)
pca1 <- dudi.pca(mafragh$xy, scan = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.label(mafragh$xy, nb = mafragh$nb, psub.text = "Samples & Neighbourhood graph", plot = FALSE)
  g2 <- s.value(mafragh$xy, coa1$li[, 1], psub.text = "Axis 1 - COA", plot = FALSE)
  g3 <- s.value(mafragh$xy, pca1$li[, 1], psub.text = "Axis 1 - PCA", plot = FALSE)
  g4 <- s.class(pca1$li, mafragh$partition, psub.text = "Plane 1-2 - PCA", plot = FALSE)
  g5 <- s.class(coa1$li, mafragh$partition, psub.text = "Plane 1-2 - COA", plot = FALSE)
  g6 <- s.class(mafragh$xy, mafragh$partition, chullSize = 1, ellipseSize = 0, starSize = 0, ppoints.cex = 0, plot = FALSE)
  G <- ADEgS(c(g1, g2, g3, g4, g5, g6), layout = c(3, 2))
} else {
  par(mfrow = c(3, 2))
  s.label(mafragh$xy, inc = FALSE, neig = mafragh$neig, sub = "Samples & Neighbourhood graph")
  s.value(mafragh$xy, coa1$li[, 1], sub = "Axis 1 - COA")
  s.value(mafragh$xy, pca1$li[, 1], sub = "Axis 1 - PCA")
  s.class(pca1$li, mafragh$partition, sub = "Plane 1-2 - PCA")
  s.class(coa1$li, mafragh$partition, sub = "Plane 1-2 - COA")
  s.chull(mafragh$xy, mafragh$partition, optchull = 1)
  par(mfrow = c(1, 1))
}

## Not run:
link1 <- area2link(mafragh$area)
neig1 <- neig(mat01 = 1*(link1 > 0))
nb1 <- neig2nb(neig1)
if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    g7 <- s.label(mafragh$xy, Sp = mafragh$Spatial, pSp.col = "white", plot = FALSE)
    g8 <- s.label(mafragh$xy, Sp = mafragh$Spatial, pSp.col = "white", nb = nb1, plab.cex = 0, pnb.node.cex = 0, ppoints.cex = 0, plot = FALSE)
    G <- ADEgS(c(g7, g8), layout = c(2, 1))
  }
} else {
  par(mfrow = c(2, 1))
  area.plot(mafragh$area, center = mafragh$xy, clab = 0.75)
  area.plot(mafragh$area, center = mafragh$xy, graph = neig1)
  par(mfrow = c(1, 1))
}

if(requireNamespace("spdep", quietly = TRUE)) {
  lw1 <- apply(link1, 1, function(x) x[x > 0])
mantel.randtest <- spdep::nb2listw(nb1, lw1)
coa1 <- dudi.coa(mafrah$flo, scan = FALSE, nf = 4)
ms1 <- multispati(coa1, listw1, scan = FALSE, nfp = 2, nfn = 0)
summary(ms1)

if(adegraphicsLoaded()) {
  if(requireNamespace("lattice", quietly = TRUE)) {
    g9 <- s1d.barchart(coa1$eig, p1d.hori = FALSE, plot = FALSE)
g10 <- s1d.barchart(ms1$eig, p1d.hori = FALSE, plot = FALSE)
g11 <- s.corcircle(ms1$as, plot = FALSE)
g12 <- lattice::xyplot(ms1$li[, 1] ~ coa1$li[, 1])
G <- ADEgS(list(g9, g10, g11, g12), layout = c(2, 2))
  }
} else {
  par(mfrow = c(2, 2))
  barplot(coa1$eig)
  barplot(ms1$eig)
  s.corcircle(ms1$as)
  plot(coa1$li[, 1], ms1$li[, 1])
  par(mfrow = c(1, 1))
}

## End(Not run)

mantel.randtest

Mantel test (correlation between two distance matrices (in C.).

Description

Performs a Mantel test between two distance matrices.

Usage

mantel.randtest(m1, m2, nrepet = 999, ...)

Arguments

m1 an object of class dist
m2 an object of class dist
nrepet the number of permutations
... further arguments passed to or from other methods

Value

an object of class randtest (randomization tests)
Author(s)
Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

References

Examples
data(yanomama)
gen <- quasieuclid(as.dist(yanomama$gen))
geo <- quasieuclid(as.dist(yanomama$geo))
plot(r1 <- mantel.randtest(geo, gen), main = "Mantel's test")
r1

mantel.rtest  Mantel test (correlation between two distance matrices (in R.).

Description
Performs a Mantel test between two distance matrices.

Usage
mantel.rtest(m1, m2, nrepet = 99, ...)

Arguments
m1  an object of class dist
m2  an object of class dist
nrepet  the number of permutations
...  further arguments passed to or from other methods

Value
an object of class rtest (randomization tests)

Author(s)
Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References
maples

**Examples**

```r
data(yanomama)
gen <- quasieuclid(as.dist(yanomama$gen))
geo <- quasieuclid(as.dist(yanomama$geo))
plot(r1 <- mantel.rtest(geo, gen), main = "Mantel's test")
r1
```

---

**maples**  
*Phylogeny and quantitative traits of flowers*

---

**Description**

This data set describes the phylogeny of 17 flowers as reported by Ackerly and Donoghue (1998). It also gives 31 traits corresponding to these 17 species.

**Usage**

```r
data(maples)
```

**Format**

`tithonia` is a list containing the 2 following objects:

- `tre` is a character string giving the phylogenetic tree in Newick format.
- `tab` is a data frame with 17 species and 31 traits

**References**


**Examples**

```r
data(maples)
phy <- newick2phylog(maples$tre)
dom <- maples$tab$Dom
bif <- maples$tab$Bif
if (requireNamespace("adephylo", quietly = TRUE) & requireNamespace("ape", quietly = TRUE)) {
  phylo <- ape::read.tree(text = maples$tre)
adephylo::orthogram(dom, tre = phylo)
adephylo::orthogram(bif, tre = phylo)
par(mfrow = c(1, 2))
dotchart.phylog(phy, dom)
dotchart.phylog(phy, bif, clabel.nodes = 0.7)
par(mfrow = c(1, 1))
plot(bif, dom, pch = 20)
abline(lm(dom ~ bif))
summary(lm(dom ~ bif))
cor.test(bif, dom)
```
marriages

Correspondence Analysis Table

Description
This array contains the socio-professional repartitions of 5850 couples.

Usage
data(mariages)

Format
The mariages data frame has 9 rows and 9 columns. The rows represent the wife’s socio-professional category and the columns the husband’s socio-professional category (1982).

Codes for rows and columns are identical: agri (Farmers), ouva (Farm workers), pat (Company directors (commerce and industry)), sup (Liberal profession, executives and higher intellectual professions), moy (Intermediate professions), emp (Other white-collar workers), ouv (Manual workers), serv (Domestic staff), aut (other workers).

Source

Examples
data(mariages)
w <- dudi.coa(mariages, scan = FALSE, nf = 3)
if(!adeGraphicsLoaded()) {
  g1 <- scatter(w, met = 1, posi = "bottomleft", plot = FALSE)
g2 <- scatter(w, met = 2, posi = "bottomleft", plot = FALSE)
g3 <- scatter(w, met = 3, posi = "bottomleft", plot = FALSE)
  g4 <- score(w, 3)
  G <- ADEgS(list(g1, g2, g3), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  scatter(w, met = 1, posi = "bottom")
  scatter(w, met = 2, posi = "bottom")
  scatter(w, met = 3, posi = "bottom")
}
### mbpcaiv

_Multiblock principal component analysis with instrumental variables_

**Description**

Function to perform a multiblock redundancy analysis of several explanatory blocks \((X_1, \ldots, X_k)\), defined as an object of class `ktab`, to explain a dependent dataset \(Y\), defined as an object of class `dudi`.

**Usage**

```r
mbpcaiv(dudiY, ktabX, scale = TRUE, option = c("uniform", "none"), scannf = TRUE, nf = 2)
```

**Arguments**

- `dudiY`: an object of class `dudi` containing the dependent variables
- `ktabX`: an object of class `ktab` containing the blocks of explanatory variables
- `scale`: logical value indicating whether the explanatory variables should be standardized
- `option`: an option for the block weighting. If `uniform`, the block weight is equal to \(1/K\) for \((X_1, \ldots, X_K)\) and to \(1\) for \(X\) and \(Y\). If `none`, the block weight is equal to the block inertia.
- `scannf`: logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: integer indicating the number of kept dimensions

**Value**

A list containing the following components is returned:

- `call`: the matching call
- `tabY`: data frame of dependent variables centered, eventually scaled (if `scale=TRUE`) and weighted (if `option="uniform"`)
- `tabX`: data frame of explanatory variables centered, eventually scaled (if `scale=TRUE`) and weighted (if `option="uniform"`)
- `TL, TC`: data frame useful to manage graphical outputs
- `nf`: numeric value indicating the number of kept dimensions
- `lw`: numeric vector of row weights
- `X.cw`: numeric vector of column weights for the explanatory dataset
- `blo`: vector of the numbers of variables in each explanatory dataset
- `rank`: maximum rank of the analysis
eig numeric vector containing the eigenvalues
lX matrix of the global components associated with the whole explanatory dataset
(scores of the individuals)
lY matrix of the components associated with the dependent dataset
Yc1 matrix of the variable loadings associated with the dependent dataset
Ti matrix containing the partial components associated with each explanatory dataset
Tli matrix containing the normalized partial components associated with each explanatory dataset
Tfa matrix containing the partial loadings associated with each explanatory dataset
cov2 squared covariance between lY and Tl1
Yco matrix of the regression coefficients of the dependent dataset onto the global components
faX matrix of the regression coefficients of the whole explanatory dataset onto the global components
XYcoef list of matrices of the regression coefficients of the whole explanatory dataset onto the dependent dataset
bip block importances for a given dimension
bipc cumulated block importances for a given number of dimensions
vip variable importances for a given dimension
vipc cumulated variable importances for a given number of dimensions

Author(s)
Stéphanie Bougeard (<stephanie.bougeard@anses.fr>) and Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

References

See Also
mbpls, testdim.multiblock, randboot.multiblock

Examples
data(chickenk)
Mortality <- chickenk[[1]]
dudiY.chick <- dudi.pca(Mortality, center = TRUE, scale = TRUE, scannf = FALSE)
ktabX.chick <- ktab.list.df(chickenk[2:5])
resmbpcaiv.chick <- mbpcaiv(dudiY.chick, ktabX.chick, scale = TRUE, option = "uniform", scannf = FALSE)
Function to perform a multiblock partial least squares (PLS) of several explanatory blocks \((X_1, \ldots, X_k)\) defined as an object of class \(ktab\), to explain a dependent dataset \(Y\) defined as an object of class \(dudi\).

**Usage**

```
mbpls(dudiY, ktabX, scale = TRUE, option = c("uniform", "none"), scannf = TRUE, nf = 2)
```

**Arguments**

- **dudiY**: an object of class \(dudi\) containing the dependent variables
- **ktabX**: an object of class \(ktab\) containing the blocks of explanatory variables
- **scale**: logical value indicating whether the explanatory variables should be standardized
- **option**: an option for the block weighting. If \("uniform\), the block weight is equal to \(1/K\) for \((X_1, \ldots, X_K)\) and to 1 for \(X\) and \(Y\). If \("none\), the block weight is equal to the block inertia
- **scannf**: logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: integer indicating the number of kept dimensions

**Value**

A list containing the following components is returned:

- **call**: the matching call
- **tabY**: data frame of dependent variables centered, eventually scaled (if \("scale=TRUE\)"") and weighted (if \("option="uniform"\)"")
- **tabX**: data frame of explanatory variables centered, eventually scaled (if \("scale=TRUE\)"") and weighted (if \("option="uniform"\)"")
- **TL, TC**: data frame useful to manage graphical outputs
- **nf**: numeric value indicating the number of kept dimensions
- **lw**: numeric vector of row weights
- **X.cw**: numeric vector of column weights for the explanatory dataset
- **blo**: vector of the numbers of variables in each explanatory dataset
- **rank**: maximum rank of the analysis
numeric vector containing the eigenvalues
matrix of the global components associated with the whole explanatory dataset (scores of the individuals)
matrix of the components associated with the dependent dataset
matrix of the variable loadings associated with the dependent dataset
squared covariance between \( Y \) and \( \mathbf{T}_X \)
matrix containing the partial loadings associated with each explanatory dataset (unit norm)
matrix containing the partial components associated with each explanatory dataset
matrix of the regression coefficients of the whole explanatory dataset onto the global components
list of matrices of the regression coefficients of the whole explanatory dataset onto the dependent dataset
block importances for a given dimension
cumulated block importances for a given number of dimensions
variable importances for a given dimension
cumulated variable importances for a given number of dimensions

**Author(s)**

Stéphanie Bougeard (<stephanie.bougeard@anses.fr>) and Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

**References**


**See Also**

`mbpls`, `testdim.multiblock`, `randboot.multiblock`

**Examples**

data(chickenk)
Mortality <- chickenk[[1]]
dudiY.chick <- dudi.pca(Mortality, center = TRUE, scale = TRUE, scannf = FALSE)
tabX.chick <- ktab.list.df(chickenk[2:5])
resmbpls.chick <- mbpls(dudiY.chick, tabX.chick, scale = TRUE, option = "uniform", scannf = FALSE)
summary(resmbpls.chick)
if(adegraphicsLoaded()) plot(resmbpls.chick)
**mcoa**  

*Multiple CO-inertia Analysis*

**Description**

performs a multiple CO-inertia analysis, using an object of class `ktab`.

**Usage**

```r
mcoa(X, option = c("inertia", "lambda1", "uniform", "internal"),
    scannf = TRUE, nf = 3, tol = 1e-07)
```

## S3 method for class `Var`

```r
mcoa
```

```r
print(x, ...)
```

## S3 method for class `Var`

```r
mcoa
```

```r
summary(object, ...)
```

## S3 method for class `Var`

```r
mcoa
```

```r
plot(x, xax = 1, yax = 2, eig.bottom = TRUE, ...)
```

**Arguments**

- **X**
  an object of class `ktab`

- **option**
  a string of characters for the weightings of the arrays options:
  - "inertia" weighting of group k by the inverse of the total inertia of the array k
  - "lambda1" weighting of group k by the inverse of the first eigenvalue of the k analysis
  - "uniform" uniform weighting of groups
  - "internal" weighting included in `X$tabw`

- **scannf**
  a logical value indicating whether the eigenvalues bar plot should be displayed

- **nf**
  if scannf FALSE, an integer indicating the number of kept axes

- **tol**
  a tolerance threshold, an eigenvalue is considered positive if it is larger than `-tol*lambda1` where `lambda1` is the largest eigenvalue.

- **x, object**
  an object of class `mcoa`

- **...**
  further arguments passed to or from other methods

- **xax, yax**
  the numbers of the x-axis and the y-axis

- **eig.bottom**
  a logical value indicating whether the eigenvalues bar plot should be added

**Value**

`mcoa` returns a list of class `mcoa` containing:

- **pseudoeig**
  a numeric vector with the all pseudo eigenvalues

- **call**
  the call-up order

- **nf**
  a numeric value indicating the number of kept axes
SynVar  a data frame with the synthetic scores
axis    a data frame with the co-inertia axes
Tli     a data frame with the co-inertia coordinates
Tl1     a data frame with the co-inertia normed scores
Tax     a data frame with the inertia axes onto co-inertia axis
Tco     a data frame with the column coordinates onto synthetic scores
TL      a data frame with the factors for Tli Tl1
TC      a data frame with the factors for Tco
T4      a data frame with the factors for Tax
lambda  a data frame with the all eigenvalues (computed on the separate analyses)
cov2    a numeric vector with the all pseudo eigenvalues (synthetic analysis)

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(friday87)
w1 <- data.frame(scale(friday87$fau, scal = FALSE))
w2 <- ktab.data.frame(w1, friday87$fau.blo, tabnames = friday87$tab.names)
mcoa1 <- mcoa(w2, "lambda1", scan = FALSE)
mcoa1
summary(mcoa1)
plot(mcoa1)

mdpcoa  Multiple Double Principal Coordinate Analysis

Description

The DPCoA analysis (see dpcoa) has been developed by Pavoine et al. (2004). It has been used in genetics for describing inter-population nucleotide diversity. However, this procedure can only be used with one locus. In order to measure and describe nucleotide diversity with more than one locus, we developed three versions of multiple DPCoA by using three ordination methods: multiple co-inertia analysis, STATIS, and multiple factorial analysis. The multiple DPCoA allows the impact of various loci in the measurement and description of diversity to be quantified and described. This method is general enough to handle a large variety of data sets. It complements existing methods such as the analysis of molecular variance or other analyses based on linkage disequilibrium measures, and is very useful to study the impact of various loci on the measurement of diversity.
Usage

mdpcoa(msamples, mdistances = NULL, method =
c("mcoa", "statis", "mfa"),
option = c("inertia", "lambda1", "uniform", "internal"),
scannf = TRUE, nf = 3, full = TRUE,
nfsep = NULL, tol = 1e-07)
kplotX.mdpcoa(object, xax = 1, yax = 2, mfrow = NULL,
which.tab = 1:length(object$nX), includepop = FALSE,
clab = 0.7, cpoi = 0.7, unique.scale = FALSE,
csub = 2, possub = "bottomright")
prep.mdpcoa(dnaobj, pop, model, ...)

Arguments

msamples A list of data frames with the populations as columns, alleles as rows and abundances as entries. All the tables should have equal numbers of columns (populations). Each table corresponds to a locus;

mdistances A list of objects of class 'dist', corresponding to the distances among alleles. The order of the loci should be the same in msamples as in mdistances;

method One of the three possibilities: "mcoa", "statis", or "mfa". If a vector is given, only its first value is considered;

option One of the four possibilities for normalizing the population coordinates over the loci: "inertia", "lambda1", "uniform", or "internal". These options are used with MCoA and MFA only;

scannf a logical value indicating whether the eigenvalues bar plots should be displayed;

nf if scannf is FALSE, an integer indicating the number of kept axes for the multiple analysis;

full a logical value indicating whether all the axes should be kept in the separated analyses (one analysis, DPCoA, per locus);

nfsep if full is FALSE, a vector indicating the number of kept axes for each of the separated analyses;

tol a tolerance threshold for null eigenvalues (a value less than tol times the first one is considered as null);

object an object of class 'mdpcoa';

xax the number of the x-axis;

yax the number of the y-axis;

mfrow a vector of the form 'c(nr,nc)', otherwise computed by as special own function 'n2mfrow';

which.tab a numeric vector containing the numbers of the loci to analyse;

includepop a logical indicating if the populations must be displayed. In that case, the alleles are displayed by points and the populations by labels;

clab a character size for the labels;

cpoi a character size for plotting the points, used with 'par("cex")'*cpoint. If zero, no points are drawn;
unique.scale  if TRUE, all the arrays of figures have the same scale;
csub     a character size for the labels of the arrays of figures used with \texttt{par("cex")]*csub';
possub   a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright");
dnaobj   a list of dna sequences that can be obtained with the function \texttt{read.dna} of the ape package;
pop      a factor that gives the name of the population to which each sequence belongs;
model    a vector giving the model to be applied for the calculations of the distances for each locus. One model should be attributed to each locus, given that the loci are in alphabetical order. The models can take the following values: "raw", "JC69", "K80" (the default), "F81", "K81", "F84", "BH87", "T92", "TN93", "GG95", "logdet", or "paralin". See the help documentation for the function \texttt{"dist.dna"} of ape for a description of the models.

...  ... further arguments passed to or from other methods

Details

An object obtained by the function 	extit{mdpcoa} has two classes. The first one is "mdpcoa" and the second is either "mcoa", or "statis", or "mfa", depending on the method chosen. Consequently, other functions already available in \textit{ade4} for displaying graphical results can be used: With MCoA, - \textit{plot.mcoa}: this function displays (1) the differences among the populations according to each locus and the compromise, (2) the projection of the principal axes of the individual analyses onto the synthetic variables, (3) the projection of the principal axes of the individual analyses onto the co-inertia axes, (4) the squared vectorial covariance among the coinertia scores and the synthetic variables; - \textit{kplot.mcoa}: this function divides previous displays (figures 1, 2, or 3 described in \textit{plot.mcoa}) by giving one plot per locus.

With STATIS, - \textit{plot.statis}: this function displays (1) the scores of each locus according to the two first eigenvectors of the matrix $R_v$, (2) the scatter diagram of the differences among populations according to the compromise, (3) the weight attributed to each locus in abscissa and the vectorial covariance among each individual analysis with the notations in the main text of the paper) and the compromise analysis in ordinates, (4) the covariance between the principal component inertia axes of each locus and the axes of the compromise space; - \textit{kplot.statis}: this function displays for each locus the projection of the principal axes onto the compromise space.

With MFA, - \textit{plot.mfa}: this function displays (1) the differences among the populations according to each locus and the compromise, (2) the projection of the principal axes of the individual analyses onto the compromise, (3) the covariance between the principal component inertia axes of each locus and the axes of the compromise space, (4) for each axis of the compromise, the amount of inertia conserved by the projection of the individual analyses onto the common space. - \textit{kplot.mfa}: this function displays for each locus the projection of the principal axes and populations onto the compromise space.

Value

The functions provide the following results:

dist.ktab      returns an object of class \texttt{dist};
Author(s)
Sandrine Pavoine <pavoine@mnhn.fr>

References


See Also
dpcoa

Examples

```r
# The functions used below require the package ape
data(rhizobium)
if (requireNamespace("ape", quietly = TRUE)) {
  dat <- prep.mdpcoa(rhizobium[[1]], rhizobium[[2]],
                      model = c("F84", "F84", "F84", "F81"), pairwise.deletion = TRUE)
sam <- dat$sam
dis <- dat$dis
  # The distances should be Euclidean.
  # Several transformations exist to render a distance object Euclidean
  # (see functions cailliez, lingoes and quasieuclid in the ade4 package).
  # Here we use the quasieuclid function.
  dis <- lapply(dis, quasieuclid)
  mdpcoa1 <- mdpcoa(sam, dis, scannf = FALSE, nf = 2)

  # Reference analysis
  plot(mdpcoa1)

  # Differences between the loci
  kplot(mdpcoa1)

  # Alleles projected on the population maps.
  kplotX.mdpcoa(mdpcoa1)
}
```

meau

Ecological Data: sites-variables, sites-species, where and when

Description
This data set contains information about sites, environmental variables and Ephemeroptera Species.
Usage

```r
data(meau)
```

Format

meau is a list of 3 components.
- **env** is a data frame with 24 sites and 10 physicochemical variables.
- **fau** is a data frame with 24 sites and 13 Ephemeroptera Species.
- **design** is a data frame with 24 sites and 2 factors.
  - season: is a factor with 4 levels = seasons.
  - site: is a factor with 6 levels = sites.

Details

Data set equivalents to `meaudret`, except that one site (6) along the Bourne (a Meaudret affluent) and one physicochemical variable - the oxygen concentration were added.

Source


Examples

```r
data(meau)
pca1 <- dudi.pca(meau$env, scan = FALSE, nf = 4)
pca2 <- bca(pca1, meau$design$season, scan = FALSE, nf = 2)
if(adegraphicsLoaded()) {
  g1 <- s.class(pca1$li, meau$design$season, psub.text = "Principal Component Analysis", plot = FALSE)
g2 <- s.class(pca2$ls, meau$design$season, psub.text = "Between seasons Principal Component Analysis", plot = FALSE)
g3 <- s.corcircle(pca1$co, plot = FALSE)
g4 <- s.corcircle(pca2$as, plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  s.class(pca1$li, meau$design$season, sub = "Principal Component Analysis")
s.class(pca2$ls, meau$design$season, sub = "Between seasons Principal Component Analysis")
s.corcircle(pca1$co)
s.corcircle(pca2$as)
  par(mfrow = c(1, 1))
}
Description

This data set contains information about sites, environmental variables and Ephemeroptera Species.

Usage

data(meaudret)

Format

meaudret is a list of 4 components.

- env is a data frame with 20 sites and 9 variables.
- fau is a data frame with 20 sites and 13 Ephemeroptera Species.
- design is a data frame with 20 sites and 2 factors.
  - season is a factor with 4 levels = seasons.
  - site is a factor with 5 levels = sites along the Meaudret river.
- spe.names is a character vector containing the names of the 13 species.

Details

Data set equivalents to meau: site (6) on the Bourne (a Meaudret affluent) and oxygen concentration were removed.

Source


Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
pca2 <- bca(pca1, meaudret$design$season, scan = FALSE, nf = 2)
if(adegraphicsLoaded()) {
  g1 <- s.class(pca1$li, meaudret$design$season,
                psub.text = "Principal Component Analysis", plot = FALSE)
  g2 <- s.class(pca2$ls, meaudret$design$season,
                psub.text = "Between dates Principal Component Analysis", plot = FALSE)
  g3 <- s.corcircle(pca1$co, plot = FALSE)
g4 <- s.corcircle(pca2$as, plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
}

mfa

Multiple Factorial Analysis

Description

performs a multiple factorial analysis, using an object of class ktab.

Usage

mfa(X, option = c("lambda1", "inertia", "uniform", "internal"),
    scannf = TRUE, nf = 3)
## S3 method for class 'mfa'
plot(x, xax = 1, yax = 2, option.plot = 1:4, ...)
## S3 method for class 'mfa'
print(x, ...)
## S3 method for class 'mfa'
summary(object, ...)

Arguments

X K-tables, an object of class ktab
option a string of characters for the weighting of arrays options:
    lambda1 weighting of group k by the inverse of the first eigenvalue of the k analysis
    inertia weighting of group k by the inverse of the total inertia of the array k
    uniform uniform weighting of groups
    internal weighting included in X$tabw
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes
x, object an object of class 'mfa'
xax, yax the numbers of the x-axis and the y-axis
option.plot an integer between 1 and 4, otherwise the 4 components of the plot are displayed
... further arguments passed to or from other methods
Value

Returns a list including:

- `tab`: a data frame with the modified array
- `rank`: a vector of ranks for the analyses
- `eig`: a numeric vector with the all eigenvalues
- `li`: a data frame with the coordinates of rows
- `TL`: a data frame with the factors associated to the rows (indicators of table)
- `co`: a data frame with the coordinates of columns
- `TC`: a data frame with the factors associated to the columns (indicators of table)
- `blo`: a vector indicating the number of variables for each table
- `lisup`: a data frame with the projections of normalized scores of rows for each table
- `link`: a data frame containing the projected inertia and the links between the arrays and the reference array

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(friday87)
w1 <- data.frame(scale(friday87$fau, scal = FALSE))
w2 <- ktab.data.frame(w1, friday87$fau.blo, tabnames = friday87$tab.names)
mfa1 <- mfa(w2, scann = FALSE)
mfa1
plot(mfa1)

data(escopage)
w <- data.frame(scale(escopage$tab))
w <- ktab.data.frame(w, escopage$blo, tabnames = escopage$tab.names)
plot(mfa(w, scann = FALSE))
Description
This data set gives genetic relationships between cattle breeds with microsatellites.

Usage
data(microsatt)

Format
microsatt is a list of 4 components.

- tab contains the allelic frequencies for 18 cattle breeds (Taurine or Zebu, French or African) and 9 microsatellites.
- loci.names is a vector of the names of loci.
- loci.eff is a vector of the number of alleles per locus.
- alleles.names is a vector of the names of alleles.

Source
Extract of data prepared by D. Laloë <ugendla@dga2.jouy.inra.fr> from data used in:

References

Examples
```r
## Not run:
data(microsatt)
fac <- factor(rep(microsatt$loci.names, microsatt$loci.eff))
w <- dudi.coa(data.frame(t(microsatt$tab)), scann = FALSE)
wit <- wca(w, fac, scann = FALSE)
microsatt.ktab <- ktab.within(wit)
```
plot(sepan(microsatt.ktab)) # 9 separated correspondence analyses
plot(mcoa(microsatt.ktab, scan = FALSE))
plot(mfa(microsatt.ktab, scan = FALSE))
plot(statis(microsatt.ktab, scan = FALSE))
## End(Not run)

mjrochet

Phylogeny and quantitative traits of teleos fishes

Description
This data set describes the phylogeny of 49 teleos fishes as reported by Rochet et al. (2000). It also gives life-history traits corresponding to these 49 species.

Usage
data(mjrochet)

Format
mjrochet is a list containing the 2 following objects:

tre  is a character string giving the phylogenetic tree in Newick format.
tab  is a data frame with 49 rows and 7 traits.

Details
Variables of mjrochet$tab are the following ones: tm (age at maturity (years)), lm (length at maturity (cm)), l05 (length at 5 per cent survival (cm)), t05 (time to 5 per cent survival (years)), fb (slope of the log-log fecundity-length relationship), fm (fecundity the year of maturity), egg (volume of eggs (mm³)).

Source
Data taken from:
Summary of data - Clupeiformes : http://www.ifremer.fr/maerha/clupe.html
Summary of data - Argentiniformes : http://www.ifremer.fr/maerha/argentin.html
Summary of data - Salmoniformes : http://www.ifremer.fr/maerha/salmon.html
Summary of data - Gadiformes : http://www.ifremer.fr/maerha/gadi.html
Summary of data - Lophiiformes : http://www.ifremer.fr/maerha/loph.html
Summary of data - Atheriniformes : http://www.ifremer.fr/maerha/ather.html
Summary of data - Perciformes : http://www.ifremer.fr/maerha/perci.html
Summary of data - Pleuronectiformes : http://www.ifremer.fr/maerha/pleuro.html
Summary of data - Scorpaeniformes : http://www.ifremer.fr/maerha/scorpa.html
Phylogenetic tree : http://www.ifremer.fr/maerha/life_history.html
References


Examples

data(mjrochet)
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab <- log(mjrochet$tab)
tab0 <- data.frame(scalewt(tab))
table.phylog(tab0, mjrochet.phy, csi = 2, clabel.r = 0.75)
if (requireNamespace("adephylo", quietly = TRUE)) {
adephylo::orthogram(tab0[,1], ortho = mjrochet.phy$Bscores)
}

mld

*Multi Level Decomposition of unidimensional data*

Description

The function mld performs an additive decomposition of the input vector x onto sub-spaces associated to an orthonormal orthobasis. The sub-spaces are defined by levels of the input factor level. The function haar2level builds the factor level such that the multi level decomposition corresponds exactly to a multiresolution analysis performed with the haar basis.

Usage

mld(x, orthobas, level, na.action = c("fail", "mean"),
plot = TRUE, dfxy = NULL, phylog = NULL, ...)
haar2level(x)

Arguments

- **x** is a vector or a time serie containing the data to be decomposed. This must be a dyadic length vector (power of 2) for the function haar2level1.
- **orthobas** is a data frame containing the vectors of the orthonormal basis.
- **level** is a factor which levels define the sub-spaces on which the function mld performs the additive decomposition.
- **na.action** if 'fail' stops the execution of the current expression when x contains any missing value. If 'mean' replaces any missing values by mean(x).
- **plot** if TRUE plot x and the components resulting from the decomposition.
- **dfxy** is a data frame with two coordinates.
- **phylog** is an object of class phylog.
- **...** further arguments passed to or from other methods.
Value

A data frame with the components resulting from the decomposition.

Author(s)

Sébastien Ollier <sebastien.ollier@u-psud.fr>

References


See Also

`gridrowcol`, `orthobasis`, `orthogram`, `mra` for multiresolution analysis with various families of wavelets

Examples

```r
## Not run:
# decomposition of a time serie
data(co2)
x <- log(co2)
orthobas <- orthobasis.line(length(x))
level<-rep("D", 467)
level[1:3]<-rep("A", 3)
level[c(77,78,79,81)]<-rep("B", 4)
level[156]<-"C"
level<-as.factor(level)
res <- mld(x, orthobas, level)
sum(scale(x, scale = FALSE) - apply(res, 1, sum))
## End(Not run)
# decomposition of a biological trait on a phylogeny
data(palm)
vfruit<-palm$traits$vfruit
vfruit<-scalewt(vfruit)
palm.phy<-newick2phylog(palm$tre)
level <- rep("F", 65)
level[c(4, 21, 3, 6, 13)] <- LETTERS[1:5]
level <- as.factor(level)
res <- mld(as.vector(vfruit), palm.phy$scores, level,
phylog = palm.phy, clabel.nod = 0.7, f.phylog=0.8,
csize = 2, clabel.row = 0.7, clabel.col = 0.7)
```
Faunistic Communities and Sampling Experiment

Description

This data set gives the abundance of 32 mollusk species in 163 samples. For each sample, 4 informations are known: the sampling sites, the seasons, the sampler types and the time of exposure.

Usage

data(mollusc)

Format

mollusc is a list of 2 objects.

  - fau is a data frame with 163 samples and 32 mollusk species (abundance).
  - plan contains the 163 samples and 4 variables.

Source


Examples

data(mollusc)
coa1 <- dudi.coa(log(mollusc$fau + 1), scannf = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  g1 <- s.class(coa1$li, mollusc$plan$site, ellipseSize = 0, starSize = 0, chullSize = 1,
               xax = 2, yax = 3, plot = FALSE)
  g2 <- s.class(coa1$li, mollusc$plan$season, ellipseSize = 0, starSize = 0, chullSize = 1,
               xax = 2, yax = 3, plot = FALSE)
  g3 <- s.class(coa1$li, mollusc$plan$method, ellipseSize = 0, starSize = 0, chullSize = 1,
               xax = 2, yax = 3, plot = FALSE)
  g4 <- s.class(coa1$li, mollusc$plan$duration, ellipseSize = 0, starSize = 0, chullSize = 1,
               xax = 2, yax = 3, plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
s.chull(coa1$li, mollusc$plan$site, 2, 3, opt = 1, cpoi = 1)
s.chull(coa1$li, mollusc$plan$season, 2, 3, opt = 1, cpoi = 1)
s.chull(coa1$li, mollusc$plan$method, 2, 3, opt = 1, cpoi = 1)
s.chull(coa1$li, mollusc$plan$duration, 2, 3, opt = 1, cpoi = 1)
  par(mfrow = c(1, 1))
}
Description

The monde84 data frame gives five demographic variables for 48 countries in the world.

Usage

data(monde84)

Format

This data frame contains the following columns:

1. pib: Gross Domestic Product
2. croipop: Growth of the population
3. morta: Infant Mortality
4. anal: Literacy Rate
5. scol: Percentage of children in full-time education

Source


Examples

data(monde84)
X <- cbind.data.frame(lpib = log(monde84$pib), monde84$croipop)
Y <- cbind.data.frame(lmorta = log(monde84$morta),
                      lanal = log(monde84$anal + 1), rscol = sqrt(100 - monde84$scol))
pcaY <- dudi.pca(Y, scan = FALSE)
pcaiv1 <- pcaiv(pcaY, X0 <- scale(X), scan = FALSE)
sum(cor(pcaiv1$l1[,1], Y0 <- scale(Y))^2)
pcaiv1$eig[1] #the same
Description

This data set gives a morphological description of 153 athletes split in five different sports.

Usage

data(morphosport)

Format

morphosport is a list of 2 objects.

- tab is a data frame with 153 athletes and 5 variables.
- sport is a factor with 6 items

Details

Variables of morphosport$tab are the following ones: dbi (biacromial diameter (cm)), tde (height (cm)), tas (distance from the buttocks to the top of the head (cm)), lms (length of the upper limbs (cm)), poids (weight (kg)).

The levels of morphosport$sport are: athl (athletics), foot (football), hand (handball), judo, nata (swimming), voll (volleyball).

Source


Examples

data(morphosport)
plot(discrimin(dudi.pca(morphosport$tab, scan = FALSE), morphosport$sport, scan = FALSE))
**Description**

Minimal Spanning Tree

**Usage**

```r
mstree(xdist, ngmax = 1)
```

**Arguments**

- `xdist`: an object of class `dist` containing an observed dissimilarity
- `ngmax`: a component number (default=1). Select 1 for getting classical MST. To add n supplementary edges k times: select k+1.

**Value**

returns an object of class `neig`

**Author(s)**

Daniel Chessel

**Examples**

```r
data(mafragh)
maf.coa <- dudi.coa(mafragh$flo, scan = FALSE)
maf.mst <- ade4::mstree(dist.dudi(maf.coa), 1)

if(adegraphicsLoaded()) {
  g0 <- s.label(maf.coa$li, plab.cex = 0, ppoints.cex = 2, nb = neig2nb(maf.mst))
} else {
  s.label(maf.coa$li, clab = 0, cpoi = 2, neig = maf.mst, cnei = 1)
}

xy <- data.frame(x = runif(20), y = runif(20))

if(adegraphicsLoaded()) {
  g1 <- s.label(xy, xlim = c(0, 1), ylim = c(0, 1),
                nb = neig2nb(ade4::mstree(dist.quant(xy, 1), 1)), plot = FALSE)
  g2 <- s.label(xy, xlim = c(0, 1), ylim = c(0, 1),
                nb = neig2nb(ade4::mstree(dist.quant(xy, 1), 2)), plot = FALSE)
  g3 <- s.label(xy, xlim = c(0, 1), ylim = c(0, 1),
                nb = neig2nb(ade4::mstree(dist.quant(xy, 1), 3)), plot = FALSE)
  g4 <- s.label(xy, xlim = c(0, 1), ylim = c(0, 1),
                nb = neig2nb(ade4::mstree(dist.quant(xy, 1), 4)), plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
}
multiblock

Display and summarize multiblock objects

Description

Generic methods print and summary for multiblock objects

Usage

## S3 method for class 'multiblock'
summary(object, ...)
## S3 method for class 'multiblock'
print(x, ...)

Arguments

- **object**: an object of class multiblock created by \texttt{mbpls} or \texttt{mbpcaiv}
- **x**: an object of class multiblock created by \texttt{mbpls} or \texttt{mbpcaiv}
- **...**: other arguments to be passed to methods

Author(s)

Stéphanie Bougeard (<stephanie.bougeard@anses.fr>) and Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

References

*Journal of Statistical Software*, 86(1), 1-17. doi: 10.18637/jss.v086.i01

See Also

\texttt{mbpls}, \texttt{mbpcaiv}
multispati

Multivariate spatial analysis

Description

These functions are deprecated. See the function multispati and the methods plot.multispati, summary.multispati and print.multispati in the package adespatial.

This function ensures a multivariate extension of the univariate method of spatial autocorrelation analysis. By accounting for the spatial dependence of data observations and their multivariate covariance simultaneously, complex interactions among many variables are analysed. Using a methodological scheme borrowed from duality diagram analysis, a strategy for the exploratory analysis of spatial pattern in the multivariate is developed.

Usage

multispati(dudi, listw, scannf = TRUE, nfposi = 2, nfnega = 0)
## S3 method for class 'multispati'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'multispati'
summary(object, ...)
## S3 method for class 'multispati'
print(x, ...)

Arguments

dudi
an object of class dudi for the duality diagram analysis
listw
an object of class listw for the spatial dependence of data observations
scannf
a logical value indicating whether the eigenvalues bar plot should be displayed
nfposi
an integer indicating the number of kept positive axes
nfnega
an integer indicating the number of kept negative axes
x, object
an object of class multispati
xax, yax
the numbers of the x-axis and the y-axis
... further arguments passed to or from other methods

Details

This analysis generalizes the Wartenberg’s multivariate spatial correlation analysis to various duality diagrams created by the functions (dudi.pca, dudi.coa, dudi.acm, dudi.mix...) If dudi is a duality diagram created by the function dudi.pca and listw gives spatial weights created by a row normalized coding scheme, the analysis is equivalent to Wartenberg’s analysis.

We note X the data frame with the variables, Q the column weights matrix and D the row weights matrix associated to the duality diagram dudi. We note L the neighbouring weights matrix associated to listw. Then, the ‘multispati’ analysis gives principal axes \( v \) that maximize the product of spatial autocorrelation and inertia of row scores:

\[
I(XQv) \cdot ||XQv||^2 = v^tQ^tX^tDLXQv
\]
Value

Returns an object of class multispati, which contains the following elements:

- **eig**: a numeric vector containing the eigenvalues
- **nfposi**: integer, number of kept axes associated to positive eigenvalues
- **nfnega**: integer, number of kept axes associated to negative eigenvalues
- **c1**: principle axes (v), data frame with p rows and (nfposi + nfnega) columns
- **l1**: principal components (XQv), data frame with n rows and (nfposi + nfnega) columns
- **ls**: lag vector onto the principal axes (LXQv), data frame with n rows and (nfposi + nfnega) columns
- **as**: principal axes of the dudi analysis (u) onto principal axes of multispati ((u)Qv), data frame with dudi$nf rows and (nfposi + nfnega) columns

Author(s)

- Daniel Chessel
- Sebastien Ollier <sebastien.ollier@u-psud.fr>
- Thibaut Jombart <t.jombart@imperial.ac.uk>

References


See Also

dudi, mat2listw

Examples

```r
## Not run:
if (requireNamespace("spdep", quietly = TRUE)) {
  data(mafragh)
  maf.xy <- mafragh$xy
  maf.flo <- mafragh$flo
```
```r
maf.listw <- spdep::nb2listw(neig2nb(mafragh$neig))
if(adegraphicsLoaded()) {
  g1 <- s.label(maf.xy, nb = neig2nb(mafragh$neig), plab.cex = 0.75)
} else {
  s.label(maf.xy, neig = mafragh$neig, clab = 0.75)
}
maf.coa <- dudi.coa(maf.flo, scannf = FALSE)
maf.coa.ms <- multispati(maf.coa, maf.listw, scannf = FALSE, nfposi = 2, nfnega = 2)
maf.coa.ms

### detail eigenvalues components
fgraph <- function(obj){
  # use multispati summary
  sum.obj <- summary(obj)
  # compute Imin and Imax
  L <- spdep::listw2mat(eval(as.list(obj$call)$listw))
  Imin <- min(eigen(0.5*(L+t(L)))$values)
  Imax <- max(eigen(0.5*(L+t(L)))$values)
  I0 <- -1/(nrow(obj$li)-1)
  # create labels
  labels <- lapply(1:length(obj$eig),function(i) bquote(lambda[.(i)]))
  # draw the plot
  xmax <- eval(as.list(obj$call)$dudi)$eig[1]*1.1
  par(las=1)
  var <- sum.obj[,2]
  Moran <- sum.obj[,3]
  plot(x=var,y=Moran,type='n',xlab='Inertia',ylab="Spatial autocorrelation (I)",
       xlim=c(0,xmax),ylim=c(Imin*1.1,Imax*1.1),yaxt='n')
  text(x=var,y=Moran,do.call(expression,labels))
  ytick <- c(I0,round(seq(Imin,Imax,le=5),1))
  ytlab <- as.character(round(seq(Imin,Imax,le=5),1))
  ytlab <- c(as.character(round(I0,1)),as.character(round(Imin,1)),
             ytlab[2:4],as.character(round(Imax,1)))
  axis(side=2,at=ytick,labels=ylab)
  rect(0,Imin,xmax,Imax,lty=2)
  segments(0,I0,xmax,I0,lty=2)
  abline(v=0)
  title("Spatial and inertia components of the eigenvalues")
}
fgraph(maf.coa.ms)

## end eigenvalues details

if(adegraphicsLoaded()) {
  g2 <- std.barchart(maf.coa$eig, p1d.hori = FALSE, plot = FALSE)
  g3 <- std.barchart(maf.coa.ms$eig, p1d.hori = FALSE, plot = FALSE)
  g4 <- s.corcircle(maf.coa.ms$as, plot = FALSE)
  G1 <- ADEgS(list(g2, g3, g4), layout = c(1, 3))
} else {
  par(mfrow = c(1, 3))
  barplot(maf.coa$eig)
  barplot(maf.coa.ms$eig)
  s.corcircle(maf.coa.ms$as)
```
multispati.randtest

Multivariate spatial autocorrelation test (in C)

Description

This function performs a multivariate autocorrelation test.
multispati.randtest

Usage

```r
multispati.randtest(dudi, listw, nrepet = 999, ...)
```

Arguments

dudi an object of class dudi for the duality diagram analysis
listw an object of class listw for the spatial dependence of data observations
nrepet the number of permutations
... further arguments passed to or from other methods

Details

We note $X$ the data frame with the variables, $Q$ the column weights matrix and $D$ the row weights matrix associated to the duality diagram `dudi`. We note $L$ the neighbouring weights matrix associated to `listw`. This function performs a Monte-Carlo Test on the multivariate spatial autocorrelation index:

$$r = \frac{\text{trace}(X^tDLXQ)}{\text{trace}(X^tDXQ)}$$

Value

Returns an object of class `randtest` (randomization tests).

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

References


See Also

dudi, mat2listw

Examples

```r
if (requireNamespace("spdep", quietly = TRUE)) {
  data(mafragh)
  maf.listw <- spdep::nb2listw(neig2nb(mafragh$neig))
  maf.pca <- dudi.pca(mafragh$env, scannf = FALSE)
  multispati.randtest(maf.pca, maf.listw)
  maf.pca.ms <- multispati(maf.pca, maf.listw, scannf = FALSE)
  plot(maf.pca.ms)
}
```
multispati.rtest  
Multivariate spatial autocorrelation test

Description
This function performs a multivariate autocorrelation test.

Usage
multispati.rtest(dudi, listw, nrepet = 99, ...)

Arguments
dudi  an object of class dudi for the duality diagram analysis
listw an object of class listw for the spatial dependence of data observations
nrepet the number of permutations
... further arguments passed to or from other methods

Details
We note X the data frame with the variables, Q the column weight matrix and D the row weight matrix associated to the duality diagram dudi. We note L the neighbouring weights matrix associated to listw. This function performs a Monte-Carlo Test on the multivariate spatial autocorrelation index:

$$r = \frac{X^tDLXQ}{X^tDXQ}$$

Value
Returns an object of class randtest (randomization tests).

Author(s)
Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

References

See Also
dudi, mat2listw
Examples

```r
if (requireNamespace("spdep", quietly = TRUE)) {
  data(mafragh)
  maf.listw <- spdep::nb2listw(neig2nb(mafragh$neig))
  maf.pca <- dudi.pca(mafragh$env, scannf = FALSE)
  multispati.rtest(maf.pca, maf.listw)
  maf.pca.ms <- multispati(maf.pca, maf.listw, scannf = FALSE)
  plot(maf.pca.ms)
}
```

Description

`neig` creates objects of class `neig` with:
- a list of edges
- a binary square matrix
- a list of vectors of neighbours
- an integer (linear and circular graphs)
- a data frame of polygons (area)

`scores.neig` returns the eigenvectors of neighbouring, orthonormalized scores (null average, unit variance 1/n and null covariances) of maximal autocorrelation.

`nb2neig` returns an object of class `neig` using an object of class `nb` in the library 'spdep'
`neig2nb` returns an object of class `nb` using an object of class `neig`
`neig2mat` returns the incidence matrix between edges (1 = neighbour; 0 = no neighbour)
`neig.util.GtoL` and `neig.util.LtoG` are utilities.

Usage

```r
neig(list = NULL, mat01 = NULL, edges = NULL, n.line = NULL, n.circle = NULL, area = NULL)
scores.neig (obj)
## S3 method for class 'neig'
print(x, ...)
## S3 method for class 'neig'
summary(object, ...)
nb2neig (nb)
neig2nb (neig)
neig2mat (neig)
```
Arguments

list  a list which each component gives the number of neighbours
mat01 a symmetric square matrix of 0-1 values
edges a matrix of 2 columns with integer values giving a list of edges
n.line the number of points for a linear plot
n.circle the number of points for a circular plot
area  a data frame containing a polygon set (see area.plot)
nb    an object of class ‘nb’
neig, x, obj, object
an object of class ‘neig’
...  further arguments passed to or from other methods

Author(s)

Daniel Chessel

References


Examples

if(!adegraphicsLoaded()) {
  if(requireNamespace("deldir", quietly = TRUE)) {
    data(mafragh)
    par(mfrow = c(2, 1))
    provi <- deldir::deldir(mafragh$xy)
    provi.neig <- neig(edges = as.matrix(provi$delsgs[, 5:6]))

    s.label(mafragh$xy, neig = provi.neig, inc = FALSE,
            addax = FALSE, clab = 0, cnei = 2)
    dist <- apply(provi.neig, 1, function(x)
                  sqrt(sum((mafragh$xy[x[1], ] - mafragh$xy[x[2], ]) ^ 2)))
    hist(dist, nclass = 50)
    mafragh.neig <- neig(edges = provi.neig[dist < 50, ])
    s.label(mafragh$xy, neig = mafragh.neig, inc = FALSE,
             addax = FALSE, clab = 0, cnei = 2)
    par(mfrow = c(1, 1))

    data(irishdata)
    irish.neig <- neig(area = irishdata$area)
    summary(irish.neig)
    print(irish.neig)
    s.label(irishdata$xy, neig = irish.neig, cneig = 3,
             area = irishdata$area, clab = 0.8, inc = FALSE)
irish.scores <- scores.neig(irish.neig)
par(mfrow = c(2, 3))
for(i in 1:6)
  s.value(irish.data$xy, irish.scores[, i], inc = FALSE, grid = FALSE, addax = FALSE,
           neig = irish.neig, csi = 2, cleg = 0, sub = paste("Eigenvector ",i), csub = 2)
par(mfrow = c(1, 1))

a.neig <- neig(n.circle = 16)
a.scores <- scores.neig(a.neig)
xy <- cbind.data.frame(cos((1:16) * pi / 8), sin((1:16) * pi / 8))
par(mfrow = c(4, 4))
for(i in 1:15)
  s.value(xy, a.scores[, i], neig = a.neig, csi = 3, cleg = 0)
par(mfrow = c(1, 1))
a.neig <- neig(n.line = 28)
a.scores <- scores.neig(a.neig)
par(mfrow = c(7, 4))
par(mar = c(1.1, 2.1, 0.1, 0.1))
for(i in 1:27)
  barplot(a.scores[, i], col = grey(0.8))
par(mfrow = c(1, 1))

if(requireNamespace("spdep", quietly = TRUE)) {
  data(mafragh)
  maf.rel <- spdep::relativeneigh(as.matrix(mafragh$xy))
  maf.rel <- spdep::graph2nb(maf.rel)
  s.label(mafragh$xy, neig = neig(list = maf.rel), inc = FALSE,
           clab = 0, addax = FALSE, cne = 1, cpo = 2)
  par(mfrow = c(2, 2))
  w <- matrix(runif(100), 50, 2)
  x.gab <- spdep::gabrielneigh(w)
  x.gab <- spdep::graph2nb(x.gab)
  s.label(data.frame(w), neig = neig(list = x.gab), inc = FALSE,
           clab = 0, addax = FALSE, cne = 1, cpo = 2, sub = "relative")
  x.rel <- spdep::relativeneigh(w)
  x.rel <- spdep::graph2nb(x.rel)
  s.label(data.frame(w), neig = neig(list = x.rel), inc = FALSE,
           clab = 0, addax = FALSE, cne = 1, cpo = 2, sub = "Gabriel")
  k1 <- spdep::knn2nb(spdep::kneighneigh(w))
  s.label(data.frame(w), neig = neig(list = k1), inc = FALSE,
           clab = 0, addax = FALSE, cne = 1, cpo = 2, sub = "k nearest neighbours")
  all.linked <- max(unlist(spdep::nbdists(k1, w)))
  z <- spdep::dneighneigh(w, 0, all.linked)
  s.label(data.frame(w), neig = neig(list = z), inc = FALSE, clab = 0,
           addax = FALSE, cne = 1, cpo = 2, sub = "Neighbourhood contiguity by distance")
  par(mfrow = c(1, 1))
}
newick.eg                 Phylogenetic trees in Newick format

Description

This data set contains various examples of phylogenetic trees in Newick format.

Usage

data(newick.eg)

Format

newick.eg is a list containing 14 character strings in Newick format.

Source

Trees 1 to 7 were obtained from the URL http://evolution.genetics.washington.edu/phylip/newicktree.html.
Trees 8 and 9 were obtained by Clémentine Carpentier-Gimaret.
Tree 10 was obtained from Treezilla Data Sets.
Trees 11 and 12 are taken from Bauwens and Díaz-Uriarte (1997).
Tree 13 is taken from Cheverud and Dow (1985).
Tree 13 is taken from Martins and Hansen (1997).

References


Examples

data(newick.eg)
newick2phylog(newick.eg[[11]])
radiial.phylog(newick2phylog(newick.eg[[7]]), circ = 1,
clabel.l = 0.75)
Description

The first three functions ensure to create object of class phylog from either a character string in Newick format (newick2phylog) or an object of class 'hclust' (hclust2phylog) or a taxonomy (taxo2phylog). The function newick2phylog.addtools is an internal function called by newick2phylog, hclust2phylog and taxo2phylog when newick2phylog.addtools = TRUE. It adds some items in 'phylog' objects.

Usage

newick2phylog(x.tre, add.tools = TRUE, call = match.call())
hclust2phylog(hc, add.tools = TRUE)
taxo2phylog(taxo, add.tools = FALSE, root="Root", abbrev=TRUE)
newick2phylog.addtools(res, tol = 1e-07)

Arguments

x.tre a character string corresponding to a phylogenetic tree in Newick format (http://evolution.genetics.washington.edu/phylip/newicktree.html)
add.tools if TRUE, executes the function newick2phylog.addtools
call call
hc an object of class hclust
taxo an object of class taxo
res an object of class phylog (an internal argument of the function newick2phylog)
tol used in case 3 of method as a tolerance threshold for null eigenvalues
root a character string for the root of the tree
abbrev logical : if TRUE levels are abbreviated by column and two characters are added before

Value

Return object of class phylog.

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

See Also

phylog, plot.phylog, as.taxo
Examples

```r
w <- "((((,,),,),(,)),,));"
w.phy <- newick2phylog(w)
print(w.phy)
plot(w.phy)

## Not run:
# newick2phylog
data(newick.eg)
radius.phylog(newick2phylog(newick.eg[[8]], FALSE), cnode = 1,
clabel.l = 0.8)

w <- NULL
w[1] <- "((,((((((((((((((,,),),),(,),),,(,),),(,),),(,),),(,),),(,),),(,),),(,),),(,),),(,).............)

```

```r
phy1 <- newick2phylog(w, FALSE)
phy1
radius.phylog(phy1, clabel.l = 0, circle = 2.2, clea = 0.5,
cnod = 0.5)
data(newick.eg)
radius.phylog(newick2phylog(newick.eg[[8]], FALSE), cnode = 1,
clabel.1 = 0.8)

# hclust2phylog
data(USArrests)
hc <- hclust(dist(USArrests), "ave")
par(mfrow = c(1,2))
plot(hc, hang = -1)

```

```r
phy <- hclust2phylog(hc)
plot(phy, clabel.1 = 0.75, clabel.n = 0.6, f = 0.75)
```

```r
```
par(mfrow = c(1,1))
row.names(USArrests)
names(phy$leaves) #WARNING not the same for two reasons
row.names(USArrests) <- gsub(" ", ",row.names(USArrests))
row.names(USArrests)
names(phy$leaves) #WARNING not the same for one reason
USArrests <- USArrests[names(phy$leaves),]
row.names(USArrests)
names(phy$leaves) #the same

table.phylog(data.frame(scalewt(USArrests)), phy, csi = 2.5,
clabel.r = 0.75, f = 0.7)
#taxo2phylog
data(taxo.eg)
tax <- as.taxo(taxo.eg[[1]])
tax.phy <- taxo2phylog(as.taxo(taxo.eg[[1]]))
par(mfrow = c(1,2))
plot(tax.phy, clabel.l = 1.25, clabel.n = 1.25, f = 0.75)
plot(taxo2phylog(as.taxo(taxo.eg[[1]])), sample(15), clabel.l = 1.25, clabel.n = 1.25, f = 0.75)
par(mfrow=c(1,1))
plot(taxo2phylog(as.taxo(taxo.eg[[2]])), clabel.l = 1, clabel.n = 0.75, f = 0.65)

## End(Not run)

---

**niche**

Method to Analyse a pair of tables: Environmental and Faunistic Data

**Description**

performs a special multivariate analysis for ecological data.

**Usage**

```r
niche(dudiX, Y, scannf = TRUE, nf = 2)
## S3 method for class 'niche'
print(x, ...)
## S3 method for class 'niche'
plot(x, xax = 1, yax = 2, ...)
niche.param(x)
## S3 method for class 'niche'
rtest(xtest, nrepet=99, ...)
```

**Arguments**

dudiX a duality diagram providing from a function dudi.coa, dudi.pca, ... using an array sites-variables
niche

Y a data frame sites-species according to dudiX$tab with no columns of zero
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes
x an object of class niche
... further arguments passed to or from other methods
xax, yax the numbers of the x-axis and the y-axis
xtest an object of class niche
nrepet the number of permutations for the testing procedure

Value Returns a list of the class niche (sub-class of dudi) containing:
rank an integer indicating the rank of the studied matrix
nf an integer indicating the number of kept axes
RV a numeric value indicating the RV coefficient
eig a numeric vector with the all eigenvalues
lw a data frame with the row weights (crossed array)
tab a data frame with the crossed array (averaging species/sites)
li a data frame with the species coordinates
l1 a data frame with the species normed scores
c0 a data frame with the variable coordinates
c1 a data frame with the variable normed scores
ls a data frame with the site coordinates
as a data frame with the axis upon niche axis

Author(s)
Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References

Examples
data(doubs)
dudi1 <- dudi.pca(doubs$env, scale = TRUE, scan = FALSE, nf = 3)
nic1 <- niche(dudi1, doubs$fish, scann = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.traject(dudi1$li, plab.cex = 0, plot = FALSE)
nipals

Non-linear Iterative Partial Least Squares (NIPALS) algorithm

Description

This function performs NIPALS algorithm, i.e. a principal component analysis of a data table that can contain missing values.

Usage

nipals(df, nf = 2, rec = FALSE, niter = 100, tol = 1e-09)
## S3 method for class 'nipals'
scatter(x, xax = 1, yax = 2, clab.row = 0.75, clab.col = 1, posieig = "top", sub = NULL, ...)

## S3 method for class 'nipals'
print(x, ...)

### Arguments
- `df`: a data frame that can contain missing values
- `nf`: an integer, the number of axes to keep
- `rec`: a logical that specify if the functions must perform the reconstitution of the data using the `nf` axes
- `niter`: an integer, the maximum number of iterations
- `tol`: a real, the tolerance used in the iterative algorithm
- `x`: an object of class `nipals`
- `xax`: the column number for the x-axis
- `yax`: the column number for the y-axis
- `clab.row`: a character size for the rows
- `clab.col`: a character size for the columns
- `posieig`: if "top" the eigenvalues bar plot is upside, if "bottom" it is downside, if "none" no plot
- `sub`: a string of characters to be inserted as legend
- `...`: further arguments passed to or from other methods

### Details
Data are scaled (mean 0 and variance 1) prior to the analysis.

### Value
Returns a list of classes `nipals`:
- `tab`: the scaled data frame
- `eig`: the pseudoeigenvalues
- `rank`: the rank of the analyzed matrice
- `nf`: the number of factors
- `c1`: the column normed scores
- `co`: the column coordinates
- `li`: the row coordinates
- `call`: the call function
- `nb`: the number of iterations for each axis
- `rec`: a data frame obtained by the reconstitution of the scaled data using the `nf` axes
Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also
dudi.pca

Examples

data(doubs)
## nipals is equivalent to dudi.pca when there are no NA
acp1 <- dudi.pca(doubs$env, scannf = FALSE, nf = 2)
nip1 <- nipals(doubs$env)

if(adegraphicsLoaded()) {
  if(requireNamespace("lattice", quietly = TRUE)) {
    g1 <- s1d.barchart(acp1$eig, psub.text = "dudi.pca", p1d.horizontal = FALSE, plot = FALSE)
    g2 <- s1d.barchart(nip1$eig, psub.text = "nipals", p1d.horizontal = FALSE, plot = FALSE)
    g3 <- lattice::xyplot(nip1$c1[, 1] ~ acp1$c1[, 1], main = "col scores", xlab = "dudi.pca", ylab = "nipals")
    g4 <- lattice::xyplot(nip1$li[, 1] ~ acp1$li[, 1], main = "row scores", xlab = "dudi.pca", ylab = "nipals")
    G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
  } else {
    par(mfrow = c(2, 2))
    barplot(acp1$eig, main = "dudi.pca")
    barplot(nip1$eig, main = "nipals")
    plot(acp1$c1[, 1], nip1$c1[, 1], main = "col scores", xlab = "dudi.pca", ylab = "nipals")
    plot(acp1$li[, 1], nip1$li[, 1], main = "row scores", xlab = "dudi.pca", ylab = "nipals")
  }
}

## Not run:
## with NAs:
doubs$env[1, 1] <- NA
nip2 <- nipals(doubs$env)
cor(nip1$li, nip2$li)
nip1$eig
nip2$eig
njplot

Phylogeny and trait of bacteria

Description

This data set describes the phylogeny of 36 bacteria as reported by Perrière and Gouy (1996). It also gives the GC rate corresponding to these 36 species.

Usage

data(njplot)

Format

njplot is a list containing the 2 following objects:

tre is a character string giving the fission tree in Newick format.
tauxcg is a numeric vector that gives the CG rate of the 36 species.

Source

Data were obtained by Manolo Gouy <manolo.gouy@univ-lyon1.fr>

References


Examples

data(njplot)
njplot.phy <- newick2phylog(njplot$tre)
par(mfrow = c(2,1))
tauxcg0 <- njplot$tauxcg - mean(njplot$tauxcg)
symbols.phylog(njplot.phy, squares = tauxcg0)
symbols.phylog(njplot.phy, circles = tauxcg0)
par(mfrow = c(1,1))
**Description**

This data set gives the performances of 33 men’s decathlon at the Olympic Games (1988).

**Usage**

```r
data(olympic)
```

**Format**

olympic is a list of 2 components.

- **tab** is a data frame with 33 rows and 10 columns events of the decathlon: 100 meters (100), long jump (long), shotput (poid), high jump (haut), 400 meters (400), 110-meter hurdles (110), discus throw (disq), pole vault (perc), javelin (jave) and 1500 meters (1500).
- **score** is a vector of the final points scores of the competition.

**Source**

Example 357 in:


**Examples**

```r
data(olympic)
pca1 <- dudi.pca(olympic$tab, scan = FALSE)

if(adegraphicsLoaded()) {
  if(requireNamespace("lattice", quietly = TRUE)) {
    g1 <- s1d.barchart(pca1$eig, p1d.hori = FALSE, plot = FALSE)
    g2 <- s.corcircle(pca1$co, plot = FALSE)
    g3 <- lattice::xyplot(pca1$l1[, 1] ~ olympic$score, type = c("p", "r"))
    g41 <- s.label(pca1$l1, plab.cex = 0.5, plot = FALSE)
    g42 <- s.arrow(2 * pca1$co, plot = FALSE)
    G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
  } else {
    par(mfrow = c(2, 2))
    barplot(pca1$eig)
    s.corcircle(pca1$co)
    plot(olympic$score, pca1$l1[, 1])
    abline(lm(pca1$l1[, 1] ~ olympic$score))
  }
}
```
This data set contains informations about environmental control and spatial structure in ecological communities of Oribatid mites.

**Usage**

data(oribatid)

**Format**

oribatid is a list containing the following objects:

- **fau**: a data frame with 70 rows (sites) and 35 columns (Oribatid species)
- **envir**: a data frame with 70 rows (sites) and 5 columns (environmental variables)
- **xy**: a data frame that contains spatial coordinates of the 70 sites

**Details**

Variables of oribatid$envir are the following ones:

- substrate: a factor with seven levels that describes the nature of the substratum
- shrubs: a factor with three levels that describes the absence/presence of shrubs
- topo: a factor with two levels that describes the microtopography
- density: substratum density ($g.L^{-1}$)
- water: water content of the substratum ($g.L^{-1}$)

**Source**

Data prepared by P. Legendre <Pierre.Legendre@umontreal.ca> and D. Borcard <borcardd@magellan.umontreal.ca>

**References**


Examples

data(oribatid)
ori.xy <- oribatid$xy[, c(2, 1)]
names(ori.xy) <- c("x", "y")
plot(ori.xy, pch = 20, cex = 2, asp = 1)

if(requireNamespace("deldir", quietly = TRUE) & requireNamespace("spdep", quietly = TRUE)) {
  plot(deldir::deldir(ori.xy), add = TRUE)
  if(adegraphicsLoaded()) {
    s.label(ori.xy, nb = spdep::knn2nb(spdep::knearneigh(as.matrix(ori.xy), 3)), plab.cex = 0)
  } else {
    s.label(ori.xy, add.p = TRUE, clab = 0,
            neig = nb2neig(spdep::knn2nb(spdep::knearneigh(as.matrix(ori.xy), 3))))
  }
}

originality

Description

computes originality values for species from an ultrametric phylogenetic tree.

Usage

originality(phyl, method = 5)

Arguments

phyl an object of class phylog
method a vector containing integers between 1 and 7.

Details

1 = Vane-Wright et al.’s (1991) node-counting index
2 = May’s (1990) branch-counting index
3 = Nixon and Wheeler’s (1991) unweighted index, based on the sum of units in binary values
4 = Nixon and Wheeler’s (1991) weighted index
5 = QE-based index
6 = Isaac et al. (2007) ED index
7 = Redding et al. (2006) Equal-split index

Value

Returns a data frame with species in rows, and the selected indices of originality in columns. Indices are expressed as percentages.

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>
References


Examples

data(carni70)
carni70.phy <- newick2phylog(carni70$tre)
ori.tab <- originality(carni70.phy, 1:7)
names(ori.tab)
dotchart.phylog(carni70.phy, ori.tab, scaling = FALSE, yjoining = 0,
           ranging = FALSE, cleaves = 0, ceti = 0.5, csub = 0.7, cdot = 0.5)

orthobasis

*Orthonormal basis for orthonormal transform*

Description

These functions returns object of class `'orthobasis'` that contains data frame defining an orthonormal basis.

`orthobasic.neig` returns the eigen vectors of the matrix N-M where M is the symmetric `n` by `n` matrix of the between-sites neighbouring graph and N is the diagonal matrix of neighbour numbers.

`orthobasis.line` returns the analytical solution for the linear neighbouring graph.

`orthobasic.circ` returns the analytical solution for the circular neighbouring graph.

`orthobasic.mat` returns the eigen vectors of the general link matrix M.

`orthobasis.haar` returns wavelet haar basis.

Usage

`orthobasis.neig(neig)`
`orthobasis.line(n)`
`orthobasis.circ(n)`
`orthobasis.mat(mat, cnw=TRUE)`
`orthobasis.haar(n)`

### S3 method for class `orthobasis'`
print(x,..., nr = 6, nc = 4)
## S3 method for class 'orthobasis'
plot(x,...)
## S3 method for class 'orthobasis'
summary(object,...)
is.orthobasis(x)

Arguments

neig is an object of class neig
n is an integer that defines length of vectors
mat is a \( n \times n \) phylogenetic or spatial link matrix
cnw if TRUE, the matrix of the neighbouring graph is modified to give Constant Neighbouring Weights
x, object is an object of class orthobasis
nr, nc the number of rows and columns to be printed
... : further arguments passed to or from other methods

Value

All the functions return an object of class orthobasis containing a data frame. This data frame defines an orthonormal basis with various attributes:

names names of the vectors
row.names row names of the data frame
class class
values optional associated eigenvalues
weights weights for the rows
call : call

Note

the function orthobasis.haar uses function \texttt{wavelet.filter} from package waveslim.

Author(s)

Sébastien Ollier <sebastien.ollier@u-psud.fr>
Daniel Chessel

References

See Also

gridrowcol that defines an orthobasis for square grid, phylog that defines an orthobasis for phylogenetic tree, orthogram and mld

Examples

# a 2D spatial orthobasis
w <- gridrowcol(8, 8)
if(adegraphicsLoaded()) {
  g1 <- s.value(w$xxy, w$orthobasis[, 1:16], pleg.drawKey = FALSE, pgri.text.cex = 0,
                ylim = c(0, 10), porigin.include = FALSE, paxes.draw = FALSE)
  g2 <- s1d.barchart(attr(w$orthobasis, "values"), pld.horizontal = FALSE,
                      labels = names(attr(w$orthobasis, "values")), plabels.cex = 0.7)
} else {
  par(mfrow = c(4, 4))
  for(k in 1:16)
    s.value(w$xxy, w$orthobasis[, k], cleg = 0, csi = 2, incl = FALSE,
            addax = FALSE, sub = k, csub = 4, ylim = c(0, 10), cgri = 0)
  par(mfrow = c(1, 1))
  barplot(attr(w$orthobasis, "values"))
}

# Haar 1D orthobasis
w <- orthobasis.haar(32)
par(mfrow = c(8, 4))
par(mar = c(0.1, 0.1, 0.1, 0.1))
for (k in 1:31) {
  plot(w[, k], type = "S", xlab = "", ylab = "", xaxt = "n",
       yaxt = "n", xaxs = "i", yaxs = "i", ylim = c(-4.5, 4.5))
  points(w[, k], type = "p", pch = 20, cex = 1.5)
}

# a 1D orthobasis
w <- orthobasis.line(n = 33)
par(mfrow = c(8, 4))
par(mar = c(0.1, 0.1, 0.1, 0.1))
for (k in 1:32) {
  plot(w[, k], type = "l", xlab = "", ylab = "", xaxt = "n",
       yaxt = "n", xaxs = "i", yaxs = "i", ylim = c(-1.5, 1.5))
  points(w[, k], type = "p", pch = 20, cex = 1.5)
}

if(adegraphicsLoaded()) {
  s1d.barchart(attr(w, "values"), pld.horizontal = FALSE, labels = names(attr(w, "values")),
                plab.cex = 0.7)
} else {
  par(mfrow = c(1, 1))
  barplot(attr(w, "values"))
}
w <- orthobasis.circ(n = 26)
#par(mfrow = c(5, 5))
#par(mar = c(0.1, 0.1, 0.1, 0.1))
#for (k in 1:25)
#   dotcircle(w[, k], xlim = c(-1.5, 1.5), cleg = 0)

par(mfrow = c(1, 1))
#barplot(attr(w, "values"))

## Not run:
# a spatial orthobasis
data(mafragh)
w <- orthobasis.neig(mafragh$neig)
if(adegraphicsLoaded()) {
   s.value(mafragh$xy, w[, 1:8], plegend.drawKey = FALSE)
s1d.barchart(attr(w, "values"), p1d.horizontal = FALSE)
} else {
   par(mfrow = c(4, 2))
   for(k in 1:8)
      s.value(mafragh$xy, w[, k], cleg = 0, sub = as.character(k), csub = 3)
   par(mfrow = c(1, 1))
   barplot(attr(w, "values"))
}

# a phylogenetic orthobasis
data(njplot)
phy <- newick2phylog(njplot$tre)
wA <- phy$Ascores
wW <- phy$Wscores
table.phylog(phylog = phy, wA, clabel.row = 0, clabel.col = 0.5)
table.phylog(phylog = phy, wW, clabel.row = 0, clabel.col = 0.5)

## End(Not run)

---

**ours**  

A table of Qualitative Variables

### Description

The `ours` (bears) data frame has 38 rows, areas of the "Inventaire National Forestier", and 10 columns.

### Usage

```r
data(ours)
```
**Format**

This data frame contains the following columns:

1. **altit**: importance of the altitudinal area inhabited by bears, a factor with levels:
   - 1 less than 50% of the area between 800 and 2000 meters
   - 2 between 50 and 70%
   - 3 more than 70%

2. **deniv**: importance of the average variation in level by square of 50 km², a factor with levels:
   - 1 less than 700 m
   - 2 between 700 and 900 m
   - 3 more than 900 m

3. **cloiso**: partitioning of the massif, a factor with levels:
   - 1 a great valley or a ridge isolates at least a quarter of the massif
   - 2 less than a quarter of the massif is isolated
   - 3 the massif has no split

4. **domain**: importance of the national forests on contact with the massif, a factor with levels:
   - 1 less than 400 km²
   - 2 between 400 and 1000 km²
   - 3 more than 1000 km²

5. **boise**: rate of afforestation, a factor with levels:
   - 1 less than 30%
   - 2 between 30 and 50%
   - 3 more than 50%

6. **hetra**: importance of plantations and mixed forests, a factor with levels:
   - 1 less than 5%
   - 2 between 5 and 10%
   - 3 more than 10% of the massif

7. **favor**: importance of favorable forests, plantations, mixed forests, fir plantations, a factor with levels:
   - 1 less than 5%
   - 2 between 5 and 10%
   - 3 more than 10% of the massif

8. **inexp**: importance of unworked forests, a factor with levels:
   - 1 less than 4%
   - 2 between 4 and 8%
   - 3 more than 8% of the total area

9. **citat**: presence of the bear before its disappearance, a factor with levels:
   - 1 no quotation since 1840
   - 2 1 to 3 quotations before 1900 and none after
   - 3 4 quotations before 1900 and none after
• 4 at least 4 quotations before 1900 and at least 1 quotation between 1900 and 1940

10. depart: district, a factor with levels:
   • AHP Alpes-de-Haute-Provence
   • AM Alpes-Maritimes
   • D Drôme
   • HP Hautes-Alpes
   • HS Haute-Savoie
   • I Isère
   • S Savoie

Source


Examples

```r
data(ours)
if(adegraphicsLoaded()) {
  sld.boxplot(dudi.acm(ours, scan = FALSE)$l1[, 1], ours)
} else {
  boxplot(dudi.acm(ours, scan = FALSE))
}
```

---

**palm**  
*Phylogenetic and quantitative traits of amazonian palm trees*

**Description**

This data set describes the phylogeny of 66 amazonian palm trees. It also gives 7 traits corresponding to these 66 species.

**Usage**

```r
data(palm)
```

**Format**

palm is a list containing the 2 following objects:

- **tre** is a character string giving the phylogenetic tree in Newick format.
- **traits** is a data frame with 66 species (rows) and 7 traits (columns).
Variables of palm$traits are the following ones:
- rord: specific richness with five ordered levels
- h: height in meter (squared transform)
- dqul: diameter at breast height in centimeter with five levels sout: subterranean, d1(0,5 cm), d2(5,15 cm), d3(15,30 cm) and d4(30,100 cm)
- vfruit: fruit volume in mm$^3$ (logged transform)
- vgrain: seed volume in mm$^3$ (logged transform)
- aire: spatial distribution area (km$^2$)
- alti: maximum altitude in meter (logged transform)

This data set was obtained by Clémentine Gimaret-Carpentier.

Examples

```r
## Not run:
data(palm)
palm.phy <- newick2phylog(palm$tre)
radial.phylog(palm.phy,clabel.l=1.25)

if (requireNamespace("adephylo", quietly = TRUE) & requireNamespace("ape", quietly = TRUE)) {
  tre <- ape::read.tree(text = palm$tre)
adephylo::orthogram(palm$traits[,4], tre)
}
dotchart.phylog(palm.phy,palm$traits[,4], clabel.l = 1, 
labels.n = palm.phy$Blabels, clabel.n = 0.75)
w <- cbind.data.frame(palm.phy$Bscores[,c(3,4,6,13,21)], 
scalewt((palm$traits[,4])))
names(w)[6] <- names(palm$traits[4])
table.phylog(w, palm.phy, clabel.r = 0.75, f = 0.5)

gearymoran(palm.phy$Amat, palm$traits[-c(1,3)])
}
## End(Not run)
```

data(pap)
Format

pap is a list containing the 2 following objects:

- **taxo** is a data frame with 39 species and 3 columns.
- **tab** is a data frame with 39 species and 4 traits.

Details

Variables of pap$tab are the following ones: genre (genus with 30 levels), famille (family with 6 levels), superfamille (superfamily with 2 levels).

Variables of pap$taxo are Group Size, Body Weight, Brain Weight, Litter Size.

Source

Data taken from the phylogenetic autocorrelation package

Examples

```r
data(pap)
taxo <- taxo2phylog(as.taxo(pap$taxo))
table.phylog(as.data.frame(scalewt(pap$tab)), taxo, csi = 2, clabel.nod = 0.6, f.phylog = 0.6)
```

---

**pcaiv**  
Principal component analysis with respect to instrumental variables

Description

performs a principal component analysis with respect to instrumental variables.

Usage

```r
pcaiv(dudi, df, scannf = TRUE, nf = 2)
## S3 method for class 'pcaiv'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'pcaiv'
print(x, ...)
## S3 method for class 'pcaiv'
summary(object, ...)
```
Arguments

dudi  a duality diagram, object of class dudi

df  a data frame with the same rows

scannf  a logical value indicating whether the eigenvalues bar plot should be displayed

nf  if scannf FALSE, an integer indicating the number of kept axes

x, object  an object of class pcaiv

xax  the column number for the x-axis

yax  the column number for the y-axis

...  further arguments passed to or from other methods

Value

returns an object of class pcaiv, sub-class of class dudi

  tab  a data frame with the modified array (projected variables)

  cw  a numeric vector with the column weights (from dudi)

  lw  a numeric vector with the row weights (from dudi)

  eig  a vector with the all eigenvalues

  rank  an integer indicating the rank of the studied matrix

  nf  an integer indicating the number of kept axes

  c1  a data frame with the Pseudo Principal Axes (PPA)

  li  a data frame dudi$ls with the predicted values by X

  co  a data frame with the inner products between the CPC and Y

  l1  data frame with the Constraint Principal Components (CPC)

  call  the matched call

  X  a data frame with the explanatory variables

  Y  a data frame with the dependant variables

  ls  a data frame with the projections of lines of dudi$tab on PPA

  param  a table containing information about contributions of the analyses: absolute (1) and cumulative (2) contributions of the decomposition of inertia of the dudi object, absolute (3) and cumulative (4) variances of the projections, the ration (5) between the cumulative variances of the projections (4) and the cumulative contributions (2), the square coefficient of correlation (6) and the eigenvalues of the pcaiv (7)

  as  a data frame with the Principal axes of dudi$tab on PPA

  fa  a data frame with the loadings (Constraint Principal Components as linear combinations of X

  cor  a data frame with the correlations between the CPC and X
pcaiv

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


Examples

```r
# example for the pcaiv
data(rhone)
pca1 <- dudi.pca(rhone$tab, scan = FALSE, nf = 3)
iv1 <- pcaiv(pca1, rhone$disch, scan = FALSE)
summary(iv1)
plot(iv1)

# example for the caiv
data(rpjdl)
millog <- log(rpjdl$mil + 1)
coa1 <- dudi.cca(rpjdl$fau, scann = FALSE)
caiv1 <- pcaiv(coa1, millog, scan = FALSE)

if(adegraphicsLoaded()) {
  G1 <- plot(caiv1)
  # analysis with c1 - as - li -ls
  # projections of inertia axes on PCAIV axes
  G2 <- s.corcircle(caiv1$as)
}
```
```r
# Species positions
g31 <- s.label(caiv$v1$c1, xax = 2, yax = 1, plab.cex = 0.5, xlim = c(-4, 4), plot = FALSE)
# Sites positions at the weighted mean of present species
g32 <- s.label(caiv$v1$ls, xax = 2, yax = 1, plab.cex = 0, plot = FALSE)
G3 <- superpose(g31, g32, plot = TRUE)

# Prediction of the positions by regression on environmental variables
G4 <- s.match(caiv$v1$ls, caiv$v1$li, xax = 2, yax = 1, plab.cex = 0.5)

# analysis with fa - l1 - co - cor
# canonical weights giving unit variance combinations
G5 <- s.arrow(caiv$v1$fa)

# sites position by environmental variables combinations
# position of species by averaging
G61 <- s.label(caiv$v1$l1, xax = 2, yax = 1, plab.cex = 0, ppoi.cex = 1.5, plot = FALSE)
G62 <- s.label(caiv$v1$co, xax = 2, yax = 1, plot = FALSE)
G6 <- superpose(g61, g62, plot = TRUE)
G7 <- s.distri(caiv$v1$l1, rpjdl$fau, xax = 2, yax = 1, ellipseSize = 0, starSize = 0.33)

# coherence between weights and correlations
G81 <- s.corcircle(caiv$v1$cor, xax = 2, yax = 1, plot = FALSE)
G82 <- s.arrow(caiv$v1$fa, xax = 2, yax = 1, plot = FALSE)
G8 <- cbindADEg(g81, g82, plot = TRUE)
}

} else {
  plot(caiv)

  # analysis with c1 - as - li - ls
  # projections of inertia axes on PCAIV axes
  s.corcircle(caiv$v1$as)

  # Species positions
  s.label(caiv$v1$c1, 2, 1, clab = 0.5, xlim = c(-4, 4))
  # Sites positions at the weighted mean of present species
  s.label(caiv$v1$ls, 2, 1, clab = 0, cpoi = 1, add.p = TRUE)

  # Prediction of the positions by regression on environmental variables
  s.match(caiv$v1$ls, caiv$v1$li, 2, 1, clab = 0.5)

  # analysis with fa - l1 - co - cor
  # canonical weights giving unit variance combinations
  s.arrow(caiv$v1$fa)

  # sites position by environmental variables combinations
  # position of species by averaging
  s.label(caiv$v1$l1, 2, 1, clab = 0, cpoi = 1.5)
  s.label(caiv$v1$co, 2, 1, add.plot = TRUE)

  s.distri(caiv$v1$l1, rpjdl$fau, 2, 1, cell = 0, csta = 0.33)
  s.label(caiv$v1$co, 2, 1, clab = 0.75, add.plot = TRUE)
```
# coherence between weights and correlations
par(mfrow = c(1, 2))
s.corcircle(caiv1$cor, 2, 1)
s.arrow(caiv1$fa, 2, 1)
par(mfrow = c(1, 1))

## S3 method for class 'pcaivortho'
summary(object, ...)

### Arguments
- **dudi**: a duality diagram, object of class `dudi`
- **df**: a data frame with the same rows
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes
- **object**: an object of class `pcaiv`
- **...**: further arguments passed to or from other methods

### Value
- an object of class 'pcaivortho' sub-class of class `dudi`
- **rank**: an integer indicating the rank of the studied matrix
- **nf**: an integer indicating the number of kept axes
- **eig**: a vector with the all eigenvalues
- **lw**: a numeric vector with the row weights (from `dudi`)
- **cw**: a numeric vector with the column weights (from `dudi`)
- **Y**: a data frame with the dependant variables
- **X**: a data frame with the explanatory variables
- **tab**: a data frame with the modified array (projected variables)
- **c1**: a data frame with the Pseudo Principal Axes (PPA)
- **as**: a data frame with the Principal axis of `dudi$tab` on PAP

---

**pcaivortho**

Principal Component Analysis with respect to orthogonal instrumental variables

**Description**

performs a Principal Component Analysis with respect to orthogonal instrumental variables.

**Usage**

```r
pcaivortho(dudi, df, scannf = TRUE, nf = 2)
```

- `## S3 method for class 'pcaivortho'
  summary(object, ...)`

**Arguments**

- **dudi**: a duality diagram, object of class `dudi`
- **df**: a data frame with the same rows
- **scannf**: a logical value indicating whether the eigenvalues bar plot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes
- **object**: an object of class `pcaiv`
- **...**: further arguments passed to or from other methods

**Value**

- an object of class 'pcaivortho' sub-class of class `dudi`
- **rank**: an integer indicating the rank of the studied matrix
- **nf**: an integer indicating the number of kept axes
- **eig**: a vector with the all eigenvalues
- **lw**: a numeric vector with the row weights (from `dudi`)
- **cw**: a numeric vector with the column weights (from `dudi`)
- **Y**: a data frame with the dependant variables
- **X**: a data frame with the explanatory variables
- **tab**: a data frame with the modified array (projected variables)
- **c1**: a data frame with the Pseudo Principal Axes (PPA)
- **as**: a data frame with the Principal axis of `dudi$tab` on PAP
pcaivortho

ls a data frame with the projection of lines of dudi$tab on PPA
li a data frame dudi$ls with the predicted values by X
ll a data frame with the Constraint Principal Components (CPC)
c a data frame with the inner product between the CPC and Y
param a data frame containing a summary

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


Examples

```r
## Not run:
data(avicmedi)
cla <- avicmedi$plan$reg:avicmedi$plan$str
# simple ordination
coa1 <- dudi.coa(avicmedi$fau, scan = FALSE, nf = 3)
# within region
w1 <- wca(coa1, avicmedi$plan$reg, scan = FALSE)
# no region the same result
pcaivnonA <- pcaivortho(coa1, avicmedi$plan$reg, scan = FALSE)
summary(pcaivnonA)
# region + strate
interAplusB <- pcaiv(coa1, avicmedi$plan, scan = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.class(coa1$li, cla, psub.text = "Sans contrainte", plot = FALSE)
g21 <- s.match(w1$li, w1$ls, plab.cex = 0, psub.text = "Intra Région", plot = FALSE)
g22 <- s.class(w1$li, cla, plot = FALSE)
g2 <- superpose(g21, g22)
g31 <- s.match(pcaivnonA$li, pcaivnonA$ls, plab.cex = 0, psub.text = "Contrainte Non A", plot = FALSE)
g32 <- s.class(pcaivnonA$li, cla, plot = FALSE)
g3 <- superpose(g31, g32)
g41 <- s.match(interAplusB$li, interAplusB$ls, plab.cex = 0, psub.text = "Contrainte A + B", plot = FALSE)
g42 <- s.class(interAplusB$li, cla, plot = FALSE)
g4 <- superpose(g41, g42)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
```
Simplified Analysis in Principal Coordinates

Description

performs a simplified analysis in principal coordinates, using an object of class `dist`.

Usage

```r
pcoscaled(distmat, tol = 1e-07)
```

Arguments

- `distmat` an object of class `dist`
- `tol` a tolerance threshold, an eigenvalue is considered as positive if it is larger than `-tol*lambda1` where `lambda1` is the largest eigenvalue

Value

returns a data frame containing the Euclidean representation of the distance matrix with a total inertia equal to 1

Author(s)

Daniel Chessel

References

Examples

```r
a <- 1 / sqrt(3) - 0.2
w <- matrix(c(0,0.8,0.8,a,0.8,0.8,a,
              0.8,0.8,0,a,a,a,0),4,4)
w <- as.dist(w)
w <- cailliez(w)
w
pcoscaled(w)
dist(pcoscaled(w)) # w
dist(pcoscaled(2 * w)) # the same
sum(pcoscaled(w)^2) # unity
```

data(pcw)

Description

Abundance of tropical trees, environmental variables and spatial coordinates for 50 sites. Data are available at [https://science.sciencemag.org/content/suppl/2002/01/24/295.5555.666.DC1](https://science.sciencemag.org/content/suppl/2002/01/24/295.5555.666.DC1) but plots from Barro Colorado Island were removed.

Usage

data(pcw)

Format

A list with 5 components.

- `spe`  Distribution of the abundances of 778 species in 50 sites
- `env`  Measurements of environmental variables for the 50 sites
- `xy`   Spatial coordinates for the sites (decimal degrees)
- `xy.utm` Spatial coordinates for the sites (UTM)
- `map`  Map of the study area stored as a SpatialPolygons object

Source


References


Examples

```r
if(adegraphicsLoaded()) {
  data(pcw)
  if(requireNamespace("spdep", quietly = TRUE)) {
    nb1 <- spdep::graph2nb(spdep::gabrielneigh(pcw$xy.utm), sym = TRUE)
    s.label(pcw$xy, nb = nb1, Sp = pcw$map)
  }
}
```

### perthi02

**Contingency Table with a partition in Molecular Biology**

Description

This data set gives the amino acids of 904 proteins distributed in three classes.

Usage

```r
data(perthi02)
```

Format

perthi02 is a list of 2 components.

- **tab** is a data frame 904 rows (proteins of 201 species) 20 columns (amino acids).
- **cla** is a factor of 3 classes of protein

The levels of perthi02$cla are:

- **cyto** (cytoplasmic proteins)
- **memb** (integral membran proteins)
- **peri** (periplasmic proteins)

Source


Examples

```r
data(perthi02)
plot(discrimin.coa(perthi02$tab, perthi02$cla, scan = FALSE))
```
Description

Create and use objects of class phylog.
phylog.extract returns objects of class phylog. It extracts sub-trees from a tree.
phylog.permut returns objects of class phylog. It creates the different representations compatible with tree topology.

Usage

### S3 method for class 'phylog'
print(x, ...)  
phylog.extract(phylog, node, distance = TRUE)  
phylog.permut(phylog, list.nodes = NULL, distance = TRUE)

Arguments

x, phylog : an object of class phylog
... : further arguments passed to or from other methods
node : a string of characters giving a node name. The functions extracts the tree rooted at this node.
distance : if TRUE, both functions retain branch lengths. If FALSE, they returns tree with arbitrary branch lengths (each branch length equals one)
list.nodes : a list which elements are vectors of string of character corresponding to direct descendants of nodes. This list defines one representation compatible with tree topology among the set of possibilities.

Value

Returns a list of class phylog:

tre : a character string of the phylogenetic tree in Newick format without branch length values
leaves : a vector which names corresponds to leaves and values gives the distance between leaves and nodes closest to these leaves
nodes : a vector which names corresponds to nodes and values gives the distance between nodes and nodes closest to these leaves
parts : a list which elements gives the direct descendants of each nodes
paths : a list which elements gives the path leading from the root to taxonomic units (leaves and nodes)
droot : a vector which names corresponds to taxonomic units and values gives distance between taxonomic units and the root
call : call

Wmat : a phylogenetic link matrix, generally called the covariance matrix. Matrix values \( Wmat_{ij} \) correspond to path length that lead from root to the first common ancestor of the two leaves \( i \) and \( j \)

Wdist : a phylogenetic distance matrix of class 'dist'. Matrix values \( Wdist_{ij} \) correspond to \( \sqrt{d_{ij}} \) where \( d_{ij} \) is the classical distance between two leaves \( i \) and \( j \)

Wvalues : a vector with the eigen values of Wmat

Wscores : a data frame with eigen vectors of Wmat. This data frame defines an orthobasis that could be used to calculate the orthonormal decomposition of a biological trait on a tree.

Amat : a phylogenetic link matrix stemed from Abouheif's test and defined in Ollier et al. (submited)

Avalues : a vector with the eigen values of Amat

Adim : number of positive eigen values

Ascores : a data frame with eigen vectors of Amat. This data frame defines an orthobasis that could be used to calculate the orthonormal decomposition of a biological trait on a tree.

Aparam : a data frame with attributes associated to nodes.

Bindica : a data frame giving for some taxonomic units the partition of leaves that is associated to its

Bscores : a data frame giving an orthobasis defined by Ollier et al. (submited) that could be used to calculate the orthonormal decomposition of a biological trait on a tree.

Bvalues : a vector giving the degree of phylogenetic autocorrelation for each vectors of Bscores (Moran’s form calculated with the matrix Wmat)

Blabels : a vector giving for each nodes the name of the vector of Bscores that is associated to its

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

References


See Also

newick2phylog, plot.phylog
Examples

```r
marthans.tre <- NULL
marthans.phylog <- newick2phylog(marthans.tre)

if(requireNamespace("ape", quietly = TRUE)) {
  marthans.phylo <- ape::read.tree(text = marthans.tre)
  par(mfrow = c(1, 2))
  plot(marthans.phylog, cnode = 3, f = 0.8, cle = 3)
  plot(marthans.phylo)
  par(mfrow = c(1, 1))
}
```

PI2newick

Import data files from Phylogenetic Independance Package

Description

This function ensures to transform a data set written for the Phylogenetic Independance package of Abouheif (1999) in a data set formatting for the functions of ade4.

Usage

```r
PI2newick(x)
```

Arguments

- **x** is a data frame that contains information on phylogeny topology and trait values

Value

Returns a list containing:

- **tre**: a character string giving the phylogenetic tree in Newick format
- **trait**: a vector containing values of the trait

Author(s)

Sébastien Ollier <sebastien.ollier@u-psud.fr>
Daniel Chessel

References

Examples

```r
x <- c(2.0266, 0.5832, 0.2460, 1.2963, 0.2460, 0.1565, -99.0000,
      14.1000, 11.2000, -99.0000, 21.3000, 27.5000, 1.0000, 2.0000,
      -1.0000, 4.0000, -1.0000, -1.0000, 3.0000, -1.0000, -1.0000,
      5.0000, -1.0000, -1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
      0.0000, 0.0000)
x <- matrix(x, nrow = 6)
x <- as.data.frame(x)
res <- PI2newick(x)
dotchart.phylog(newick2phylog(res$tre), res$trait)
```

piosphere

---

Plant traits response to grazing

Description

Plant species cover, traits and environmental parameters recorded around livestock watering points in different habitats of central Namibian farmlands. See the Wesuls et al. (2012) paper for a full description of the data set.

Usage

data(piosphere)

Format

- **piosphere** is a list of 4 components.
  - **veg** is a data frame containing plant species cover
  - **traits** is a data frame with plant traits
  - **env** is a data frame with environmental variables
  - **habitat** is a factor describing habitat/years for each site

Source


Examples

data(piosphere)
names(piosphere)
afcL <- dudi.coa(log(piosphere$veg + 1), scannf = FALSE)
acpR <- dudi.pca(piosphere$env, scannf = FALSE, row.w = afcL$lw)
acpQ <- dudi.hillsmith(piosphere$traits, scannf = FALSE, row.w = afcL$lw)
rlq1 <- rlq(acpR, afcL, acpQ, scannf = FALSE)
plot(rlq1)
plot.phylog  
Plot phylogenies

Description

plot.phylog draws phylogenetic trees as linear dendograms.
radial.phylog draws phylogenetic trees as circular dendograms.
enum.phylog enumerate all the possible representations for a phylogeny.

Usage

## S3 method for class 'phylog'
plot(x, y = NULL, f.phylog = 0.5, cleaves = 1, cnodes = 0,  
labels.leaves = names(x$leaves), clabel.leaves = 1,  
labels.nodes = names(x$nodes), clabel.nodes = 0, sub = "",  
csub = 1.25, possub = "bottomleft", draw.box = FALSE, ...)  
radial.phylog(phylog, circle = 1, cleaves = 1, cnodes = 0,  
labels.leaves = names(phylog$leaves), clabel.leaves = 1,  
labels.nodes = names(phylog$nodes), clabel.nodes = 0,  
draw.box = FALSE)  
enum.phylog(phylog, no.over = 1000)

Arguments

x, phylog      an object of class phylog
y              a vector which values correspond to leaves positions
f.phylog       a size coefficient for tree size (a parameter to draw the tree in proportion to leaves label)
circle         a size coefficient for the outer circle
cleaves        a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
cnodes         a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
labels.leaves  a vector of strings of characters for the leaves labels
clabel.leaves  a character size for the leaves labels, used with par("cex")*clabel.leaves. If zero, no leaves labels are drawn
labels.nodes   a vector of strings of characters for the nodes labels
clabel.nodes   a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
sub             a string of characters to be inserted as legend
csub            a character size for the legend, used with par("cex")*csub
possub          a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
`draw.box` if TRUE draws a box around the current plot with the function `box()`.

... further arguments passed to or from other methods.

`no.over` a size coefficient for the number of representations.

**Details**

The vector `y` is an argument of the function `plot.phylog` that ensures to plot one of the possible representations of a phylogeny. The vector `y` is a permutation of the set of leaves \{1,2,...,f\} compatible with the phylogeny’s topology.

**Value**

The function `enum.phylog` returns a matrix with as many columns as leaves. Each row gives a permutation of the set of leaves \{1,2,...,f\} compatible with the phylogeny’s topology.

**Author(s)**

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

**See Also**

`phylog`

**Examples**

```r
data(newick.eg)
par(mfrow = c(3,2))
for(i in 1:6) plot(newick2phylog(newick.eg[[i]], FALSE),
  clea = 2, clabel.l = 3, cnod = 2.5)
par(mfrow = c(1,1))
## Not run:
par(mfrow = c(1,2))
plot(newick2phylog(newick.eg[[11]], FALSE), clea = 1.5,
  clabel.l = 1.5, clabel.nod = 0.75, f = 0.8)
plot(newick2phylog(newick.eg[[10]], FALSE), clabel.l = 0,
  clea = 0, cn = 0, f = 1)
par(mfrow = c(1,1))
## End(Not run)
par(mfrow = c(2,2))
w7 <- newick2phylog("(((1,2,3)b),(6)c),(4,5)d,7)f;")
plot(w7,clabel.l = 1.5, clabel.n = 1.5, f = 0.8, cle = 2,
  cnod = 3, sub = "(((1,2,3)b),(6)c),(4,5)d,7)f;", csub = 2)
w <- NULL
plot(newick2phylog(w), f = 0.8, cnod = 2, cleav = 2, clabel.l = 2)
```
Results of the French presidential elections of 2002

```r
data(taxo.eg)
w <- taxo2phylog(as.taxo(taxo.eg[[1]]))
plot(w, clabel.lea = 1.25, clabel.n = 1.25, sub = "Taxonomy",
csub = 3, f = 0.8, possub = "topleft")

provi.tre <- "(((a,b,c,d,e)A,(f,g,h)B)C)D;"
provi.phy <- newick2phylog(provi.tre)
plot(provi.phy, clabel.l = 2, clabel.n = 2, f = 0.8)
par(mfrow = c(1,1))

## Not run:
par(mfrow = c(3,3))
for (j in 1:6) radial.phylog(newick2phylog(newick.eg[[j]], FALSE), clabel.l = 2, cnodes = 2)
radial.phylog(newick2phylog(newick.eg[[7]],FALSE), clabel.l = 2)
radial.phylog(newick2phylog(newick.eg[[8]],FALSE), clabel.l = 0,
circle = 1.8)
radial.phylog(newick2phylog(newick.eg[[9]],FALSE), clabel.l = 1,
clabel.n = 1, cle = 0, cnode = 1)
par(mfrow = c(1,1))

data(bsetal97)
bsetal.phy = taxo2phylog(as.taxo(bsetal97$taxo[,1:3]), FALSE)
radial.phylog(bsetal.phy, cnod = 1, clea = 1, clabel.l = 0.75,
draw.box = TRUE, cir = 1.1)
par(mfrow = c(1,1))

## End(Not run)

## Not run:
# plot all the possible representations of a phylogenetic tree
a <- "((a,b)A,(c,d,(e,f)B)C)D;"
wa <- newick2phylog(a)
wx <- enum.phylog(wa)
dim(wx)

par(mfrow = c(6,8))
fun <- function(x) {
w <-NULL
lapply(x, function(y) w <<-paste(w,as.character(y),sep=""))
plot(wa, x, clabel.n = 1.25, f = 0.75, clabel.l = 2,
box = FALSE, cle = 1.5, sub = w, csub = 2)
invisible()}
apply(wx,1,fun)
par(mfrow = c(1,1))

## End(Not run)
```
Description

presid2002 is a list of two data frames tour1 and tour2 with 93 rows (93 departments from continental Metropolitan France) and, 4 and 12 variables respectively.

Usage

data(presid2002)

Format

tour1 contains the following arguments:
the number of registered voters (inscrits); the number of abstentions (abstentions); the number of voters (votants); the number of expressed votes (exprimes) and, the numbers of votes for each candidate: Megret, Lepage, Gluksten, Bayrou, Chirac, Le_Pen, Taubira, Saint.josse, Mamere, Jospin, Boutin, Hue, Chevenement, Madelin, Besancenot.

tour2 contains the following arguments:
the number of registered voters (inscrits); the number of abstentions (abstentions); the number of voters (votants); the number of expressed votes (exprimes) and, the numbers of votes for each candidate: Chirac and Le_Pen.

Source

Site of the ministry of the Interior, of the Internal Security and of the local liberties

See Also

This dataset is compatible with elec88 and cnc2003

Examples

data(presid2002)

## Not run:
data(elec88)
data(cnc2003)
w0 <- ade4::area.util.class(elec88$area, cnc2003$reg)
w1 <- scale(elec88$tab$Chirac)
w2 <- scale(presid2002$tour1$Chirac / presid2002$tour1$exprimes)
w3 <- scale(elec88$tab$Mitterand)
w4 <- scale(presid2002$tour2$Chirac / presid2002$tour2$exprimes)

if(adegraphicsLoaded()) {
g1 <- s.value(elec88$xy, w1, Sp = elec88$Spatial, pSp.col = "white", pgrid.draw = FALSE, psub.text = "Chirac 1988 T1", plot = FALSE)
g2 <- s.value(elec88$xy, w2, Sp = elec88$Spatial, pSp.col = "white", pgrid.draw = FALSE, psub.text = "Chirac 2002 T1", plot = FALSE)
}
procella

Phylogeny and quantitative traits of birds

Description

This data set describes the phylogeny of 19 birds as reported by Bried et al. (2002). It also gives 6 traits corresponding to these 19 species.

Usage

data(procella)

Format

procella is a list containing the 2 following objects:

tre is a character string giving the phylogenetic tree in Newick format.

traits is a data frame with 19 species and 6 traits
Details

Variables of procella$traits are the following ones:
- site.fid: a numeric vector that describes the percentage of site fidelity
- mate.fid: a numeric vector that describes the percentage of mate fidelity
- mass: an integer vector that describes the adult body weight (g)
- ALE: a numeric vector that describes the adult life expectancy (years)
- BF: a numeric vector that describes the breeding frequencies
- col.size: an integer vector that describes the colony size (no nests monitored)

References


Examples

data(procella)
pro.phy <- newick2phylog(procella$tre)
plot(pro.phy, clabel.n = 1, clabel.l = 1)
wtsite.fid[is.na(wt$site.fid)] <- mean(wt$site.fid[!is.na(wt$site.fid)])
wtsite.fid <- asin(sqrt(wt$site.fid/100))
wt$ALE[is.na(wt$ALE)] <- mean(wt$ALE[!is.na(wt$ALE)])
wt$ALE <- sqrt(wt$ALE)
wt$BF[is.na(wt$BF)] <- mean(wt$BF[!is.na(wt$BF)])
wttmass <- log(wt$mass)
wtsite.fid <- wt[, -6]
table.phylog(scalewt(wtsite.fid), pro.phy, csi = 2)
gearymoran(pro.phy$Amat, wt, 9999)

procuste

Simple Procruste Rotation between two sets of points

Description

performs a simple procruste rotation between two sets of points.

Usage

procuste(dfX, dfY, scale = TRUE, nf = 4, tol = 1e-07)
## S3 method for class 'procuste'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'procuste'
print(x, ...)
## S3 method for class 'procuste'
randtest(xtest, nrepet = 999, ...)
**Arguments**

- `dfX`, `dfY`: two data frames with the same rows
- `scale`: a logical value indicating whether a transformation by the Gower’s scaling (1971) should be applied
- `nf`: an integer indicating the number of kept axes
- `tol`: a tolerance threshold to test whether the distance matrix is Euclidean: an eigenvalue is considered positive if it is larger than `-tol*lambda1` where `lambda1` is the largest eigenvalue.
- `x`, `xtest`: an object of class `procuste`
- `xax`: the column number for the x-axis
- `yax`: the column number for the y-axis
- `nrepet`: the number of repetitions to perform the randomization test
- `...`: further arguments passed to or from other methods

**Value**

returns a list of the class `procuste` with 9 components

- `d`: a numeric vector of the singular values
- `rank`: an integer indicating the rank of the crossed matrix
- `nf`: an integer indicating the number of kept axes
- `tabX`: a data frame with the array X, possibly scaled
- `tabY`: a data frame with the array Y, possibly scaled
- `rotX`: a data frame with the result of the rotation from array X to array Y
- `rotY`: a data frame with the result of the rotation from array Y to array X
- `loadX`: a data frame with the loadings of array X
- `loadY`: a data frame with the loadings of array Y
- `scorX`: a data frame with the scores of array X
- `scorY`: a data frame with the scores of array Y
- `call`: a call order of the analysis

**Author(s)**

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
References


Examples

data(macaca)
pro1 <- procuste(macaca$xy1, macaca$xy2, scal = FALSE)
pro2 <- procuste(macaca$xy1, macaca$xy2)
if(adegraphicsLoaded()) {
  g1 <- s.match(pro1$tabX, pro1$rotY, plab.cex = 0.7, plot = FALSE)
g2 <- s.match(pro1$tabY, pro1$rotX, plab.cex = 0.7, plot = FALSE)
g3 <- s.match(pro2$tabX, pro2$rotY, plab.cex = 0.7, plot = FALSE)
g4 <- s.match(pro2$tabY, pro2$rotX, plab.cex = 0.7, plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  s.match(pro1$tabX, pro1$rotY, clab = 0.7)
s.match(pro1$tabY, pro1$rotX, clab = 0.7)
s.match(pro2$tabX, pro2$rotY, clab = 0.7)
s.match(pro2$tabY, pro2$rotX, clab = 0.7)
  par(mfrow = c(1,1))
}

data(doubs)
pca1 <- dudi.pca(doubs$env, scal = TRUE, scann = FALSE)
pca2 <- dudi.pca(doubs$fish, scal = FALSE, scann = FALSE)
pro3 <- procuste(pca1$tab, pca2$tab, nf = 2)
if(adegraphicsLoaded()) {
  g11 <- s.traject(pro3$scorX, plab.cex = 0, plot = FALSE)
g12 <- s.label(pro3$scorX, plab.cex = 0.8, plot = FALSE)
g1 <- superpose(g11, g12)
g21 <- s.traject(pro3$scorY, plab.cex = 0, plot = FALSE)
g22 <- s.label(pro3$scorY, plab.cex = 0.8, plot = FALSE)
g2 <- superpose(g21, g22)
g3 <- s.arrow(pro3$loadX, plab.cex = 0.75, plot = FALSE)
g4 <- s.arrow(pro3$loadY, plab.cex = 0.75, plot = FALSE)
G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
}
procuste.randtest

Monte-Carlo Test on the sum of the singular values of a procustean rotation (in C).

Description

performs a Monte-Carlo Test on the sum of the singular values of a procustean rotation.

Usage

procuste.randtest(df1, df2, nrepet = 999, ...)

Arguments

df1 a data frame
df2 a data frame
nrepet the number of permutations
... further arguments passed to or from other methods

Value

returns a list of class randtest

Author(s)

Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

References

procuste.rtest

Examples

```r
data(dous)
pca1 <- dudi.pca(dous$env, scal = TRUE, scann = FALSE)
pca2 <- dudi.pca(dous$fish, scal = FALSE, scann = FALSE)
protest1 <- procuste.randtest(pca1$tab, pca2$tab, 999)
plot(protest1, main = "PROTEST")
```

procuste.rtest  

*Monte-Carlo Test on the sum of the singular values of a procustean rotation (in R).*

Description

performs a Monte-Carlo Test on the sum of the singular values of a procustean rotation.

Usage

```r
procuste.rtest(df1, df2, nrepet = 99, ...)
```

Arguments

- `df1`: a data frame
- `df2`: a data frame
- `nrepet`: the number of permutations
- `...`: further arguments passed to or from other methods

Value

returns a list of class `rtest`

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References

Examples

data(doubs)
pca1 <- dudi.pca(doubs$env, scal = TRUE, scann = FALSE)
pca2 <- dudi.pca(doubs$fish, scal = FALSE, scann = FALSE)
proc1 <- procuste(pca1$tab, pca2$tab)
protest1 <- procuste.rtest(pca1$tab, pca2$tab, 999)
protest1
plot(protest1)

pta

Partial Triadic Analysis of a K-tables

Description

performs a partial triadic analysis of a K-tables, using an object of class ktab.

Usage

pta(X, scannf = TRUE, nf = 2)
## S3 method for class 'pta'
plot(x, xax = 1, yax = 2, option = 1:4, ...)
## S3 method for class 'pta'
print(x, ...)

Arguments

X an object of class ktab where the arrays have 1) the same dimensions 2) the same names for columns 3) the same column weightings
scannf a logical value indicating whether the eigenvalues bar plot should be displayed
nf if scannf FALSE, an integer indicating the number of kept axes
x an object of class 'pta'
xax, yax the numbers of the x-axis and the y-axis
option an integer between 1 and 4, otherwise the 4 components of the plot are displayed
... further arguments passed to or from other methods

Value

returns a list of class 'pta', sub-class of 'dudi' containing :

RV a matrix with the all RV coefficients
RV.eig a numeric vector with the all eigenvalues (interstructure)
RV.coo a data frame with the scores of the arrays
tab.names a vector of characters with the array names
nf an integer indicating the number of kept axes
rank an integer indicating the rank of the studied matrix

tabw a numeric vector with the array weights
cw a numeric vector with the column weights
lw a numeric vector with the row weights
eig a numeric vector with the all eigenvalues (compromis)
cos2 a numeric vector with the $\cos^2$ between compromise and arrays

tab a data frame with the modified array
li a data frame with the row coordinates
l1 a data frame with the row normed scores
co a data frame with the column coordinates
c1 a data frame with the column normed scores
Tli a data frame with the row coordinates (each table)
Tco a data frame with the column coordinates (each table)
Tcomp a data frame with the principal components (each table)
Tax a data frame with the principal axes (each table)
TL a data frame with the factors for Tli
TC a data frame with the factors for Tco
T4 a data frame with the factors for Tax and Tcomp

Author(s)

Pierre Bady <pierre.bady@univ-lyon1.fr>
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(meaudret)
wit1 <- withinpca(meaudret$env, meaudret$design$season, scan = FALSE, scal = "partial")
ktal1 <- ktab.within(wit1, colnames = rep(c("S1", "S2", "S3", "S4", "S5"), 4))
ktal2 <- t(ktal1)
ptal1 <- pta(ktal2, scann = FALSE)
ptal1
plot(ptal1)
quasieuclid  

Transformation of a distance matrice to a Euclidean one

Description

transforms a distance matrix in a Euclidean one.

Usage

quasieuclid(distmat)

Arguments

distmat  
an object of class dist

Details

The function creates a distance matrice with the positive eigenvalues of the Euclidean representation.
Only for Euclidean distances which are not Euclidean for numeric approximations (for examples, in papers as the following example).

Value

object of class dist containing a Euclidean distance matrice

Author(s)

Daniel Chessel  
Stéphane Dray <stephane.dray@univ-lyon1.fr>

Examples

data(yanomama)  
geo <- as.dist(yanomama$geo)  
is.euclid(geo) # FALSE  
geo1 <- quasieuclid(geo)  
is.euclid(geo1) # TRUE  
par(mfrow = c(2,2))  
lapply(yanomama, function(x) plot(as.dist(x), quasieuclid(as.dist(x))))  
par(mfrow = c(1,1))
Description

Functions and classes to manage outputs of bootstrap simulations for one (class `randboot`) or several (class `krandboot`) statistics.

Usage

```r
as.krandboot(obs, boot, quantiles = c(0.025, 0.975), names = colnames(boot), call = match.call())
## S3 method for class 'krandboot'
print(x, ...)
```

```r
as.randboot(obs, boot, quantiles = c(0.025, 0.975), call = match.call())
## S3 method for class 'randboot'
print(x, ...)
```

```r
randboot(object, ...)
```

Arguments

- `obs` a value (class `randboot`) or a vector (class `krandboot`) with observed statistics.
- `boot` a vector (class `randboot`) or a matrix (class `krandboot`) with the bootstrap values of the statistics.
- `quantiles` a vector indicating the lower and upper quantiles to compute.
- `names` a vector of names for the statistics.
- `call` the matching call.
- `x` an object of class `randboot` or `krandboot`.
- `object` an object on which bootstrap should be performed.
- `...` other arguments to be passed to methods.

Value

an object of class `randboot` or `krandboot`.

Author(s)

Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

References

See Also

`randboot.multiblock`

Examples

```r
## an example corresponding to 10 statistics and 100 repetitions
bt <- as.krandboot(obs = rnorm(10), boot = matrix(rnorm(1000), nrow = 100))
bt
if(adegraphicsLoaded())
  plot(bt)
```

`randboot.multiblock`  
**Bootstraped simulations for multiblock methods**

Description

Function to perform bootstrapped simulations for multiblock principal component analysis with instrumental variables or multiblock partial least squares, in order to get confidence intervals for some parameters, *i.e.*, regression coefficients, variable and block importances.

Usage

```r
## S3 method for class 'multiblock'
randboot(object, nrepet = 199, optdim, ...)
```

Arguments

- `object`: an object of class multiblock created by `mbpls` or `mbpcaiv`
- `nrepet`: integer indicating the number of repetitions
- `optdim`: integer indicating the optimal number of dimensions, *i.e.*, the optimal number of global components to be introduced in the model
- `...`: other arguments to be passed to methods

Value

A list containing objects of class `krandboot`

Author(s)

Stéphanie Bougeard (<stephanie.bougeard@anses.fr>) and Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

References


See Also

`mbpcaiv`, `mbpls.testdim.multiblock`, `as.krandboot`

Examples

data(chickenk)
Mortality <- chickenk[[1]]
dudiY.chick <- dudi.pca(Mortality, center = TRUE, scale = TRUE, scannf = FALSE)
ktabX.chick <- ktab.list.df(chickenk[2:5])
resmbpcaiv.chick <- mbpcaiv(dudiY.chick, ktabX.chick, scale = TRUE,
option = "uniform", scannf = FALSE, nf = 4)
## nrepet should be higher for a real analysis

test <- randboot(resmbpcaiv.chick, optdim = 4, nrepet = 10)
test
if(adegraphicsLoaded())
plot(test$bipc)

**randtest**

Class of the Permutation Tests (in C).

Description

`randtest` is a generic function. It proposes methods for the following objects between, discrimin, coinertia ...

Usage

`randtest(xtest, ...)`
as.randtest(sim, obs, alter = c("greater", "less", "two-sided"),
output = c("light", "full"), call = match.call(), subclass = NULL)

## S3 method for class 'randtest'
plot(x, nclass = 10, coeff = 1, ...)
## S3 method for class 'randtest'
print(x, ...)

Arguments

- `xtest` an object used to select a method
- `...` further arguments passed to or from other methods; in `plot.randtest` to `hist`
- `output` a character string specifying if all simulations should be stored ("full"). This was the default until ade4 1.7-5. Now, by default ("light"), only the distribution of simulated values is stored in element `plot` as produced by the `hist` function.
- `nclass` a number of intervals for the histogram. Ignored if object `output` is "light"
coeff to fit the magnitude of the graph. Ignored if object output is "light"

sim a numeric vector of simulated values

obs a numeric vector of an observed value

alter a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two-sided"

call a call order

subclass a character vector indicating the subclasses associated to the returned object

Details

If the alternative hypothesis is "greater", a p-value is estimated as: (number of random values equal to or greater than the observed one + 1)/(number of permutations + 1). The null hypothesis is rejected if the p-value is less than the significance level. If the alternative hypothesis is "less", a p-value is estimated as: (number of random values equal to or less than the observed one + 1)/(number of permutations + 1). Again, the null hypothesis is rejected if the p-value is less than the significance level. Lastly, if the alternative hypothesis is "two-sided", the estimation of the p-value is equivalent to the one used for "greater" except that random and observed values are firstly centered (using the average of random values) and secondly transformed to their absolute values. Note that this is only suitable for symmetric random distribution.

Value

as.randtest returns a list of class randtest.
plot.randtest draws the simulated values histograms and the position of the observed value.

See Also

mantel.randtest, procuste.randtest, rtest

Examples

par(mfrow = c(2,2))
for (x0 in c(2.4,3.4,5.4,20.4)) {
  l0 <- as.randtest(sim = rnorm(200), obs = x0)
  print(l0)
  plot(l0,main=paste("p.value = ", round(l0$pvalue, dig = 5)))
}
par(mfrow = c(1,1))

Permutation tests on an analysis of molecular variance (in C).

Description

Tests the components of covariance with permutation processes described by Excoffier et al. (1992).


**randtest.between**

Monte-Carlo Test on the between-groups inertia percentage (in C).

**Description**

Performs a Monte-Carlo test on the between-groups inertia percentage.

**Usage**

```r
## S3 method for class 'between'
randtest(xtest, nrepet = 999, ...)
```

**Arguments**

- `xtest`: an object of class `amova`
- `nrepet`: the number of permutations
- `...`: further arguments passed to or from other methods

**Value**

returns an object of class `krandtest` or `randtest`

**Author(s)**

Sandrine Pavoine <pavoine@mnhn.fr>

**References**


**Examples**

```r
data(humDNAm)
amovahum <- amova(humDNAm$samples, sqrt(humDNAm$distances), humDNAm$structures)
amovahum
randtesthum <- randtest(amovahum, 49)
plot(randtesthum)
```
**Arguments**

- **xtest**: an object of class `between`
- **nrepet**: the number of permutations
- **...**: further arguments passed to or from other methods

**Value**

a list of the class `randtest`

**Author(s)**

Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

**References**


**Examples**

```r
data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 3)
rand1 <- randtest(bca(pca1, meaudret$design$season, scan = FALSE), 99)
rand1
plot(rand1, main = "Monte-Carlo test")
```

**Description**

Performs a Monte-Carlo test on a Co-inertia analysis.

**Usage**

```r
## S3 method for class 'coinertia'
randtest(xtest, nrepet = 999, fixed=0, ...)
```

**Arguments**

- **xtest**: an object of class `coinertia`
- **nrepet**: the number of permutations
- **fixed**: when non uniform row weights are used in the coinertia analysis, this parameter must be the number of the table that should be kept fixed in the permutations
- **...**: further arguments passed to or from other methods
Value

a list of the class randtest

Note

A testing procedure based on the total coinertia of the analysis is available by the function randtest.coinertia. The function allows to deal with various analyses for the two tables. The test is based on random permutations of the rows of the two tables. If the row weights are not uniform, mean and variances are recomputed for each permutation (PCA); for MCA, tables are recentred and column weights are recomputed. If weights are computed using the data contained in one table (e.g. COA), you must fix this table and permute only the rows of the other table. The case of decentred PCA (PCA where centers are entered by the user) is not yet implemented. If you want to use the testing procedure for this case, you must firstly center the table and then perform a non-centered PCA on the modified table. The case where one table is treated by hill-smith analysis (mix of quantitative and qualitative variables) will be soon implemented.

Author(s)

Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr> modified by Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


Examples

data(doubs)
dudi1 <- dudi.pca(doubs$env, scale = TRUE, scan = FALSE, nf = 3)
dudi2 <- dudi.pca(doubs$fish, scale = FALSE, scan = FALSE, nf = 2)
coin1 <- coinertia(dudi1, dudi2, scan = FALSE, nf = 2)
plot(randtest(coin1))

### randtest.discrimin

Monte-Carlo Test on a Discriminant Analysis (in C).

**Description**

Test of the sum of a discriminant analysis eigenvalues (divided by the rank). Non parametric version of the Pillai’s test. It authorizes any weighting.

**Usage**

```r
## S3 method for class 'discrimin'
randtest(xtest, nrepet = 999, ...)
```
Arguments

- xtest an object of class discrimin
- nrepet the number of permutations
- ... further arguments passed to or from other methods

Value

returns a list of class randtest

Author(s)

Jean Thioulouse <Jean.Thioulouse@univ-lyon1.fr>

Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 3)
rand1 <- randtest(discrimin(pca1, meaudret$design$season, scan = FALSE), 99)
rand1

# Monte-Carlo test
# Observation: 0.3035
# Call: as.randtest(sim = sim, obs = obs)
# Based on 999 replicates
# Simulated p-value: 0.001
plot(rand1, main = "Monte-Carlo test")
summary.manova(manova(as.matrix(meaudret$env)~meaudret$design$season), "Pillai")
  # Df Pillai approx F num Df den Df  Pr(>F)
  # meaudret$design$season 3 2.73 11.30  27  30 1.6e-09 ***
  # Residuals          16
  # ---
  # Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `..' 0.1 ` ' 1
  # 2.731/9 = 0.3034

randtest.dpcoa Permutation test for double principal coordinate analysis (DPCoA)

Description

randtest.dpcoa calculates the ratio of beta to gamma diversity associated with DPCoA and compares the observed value to values obtained by permuting data.

Usage

## S3 method for class 'dpcoa'
randtest(xtest, model = c("lp","ls"), nrepet = 99, alter = c("greater", "less", "two-sided"), ...)
Arguments

xtest an object of class dpcoa
model either "1p", "1s", or the name of a function, (see details)
nrepet the number of permutations to perform, the default is 99
alter a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two-sided"
... further arguments passed to or from other methods

Details

Model 1p permutes the names of the columns of the abundance matrix. Model 1s permutes the abundances of the categories (columns of the abundance matrix, usually species) within collections (rows of the abundance matrix, usually communities). Only the categories with positive abundances are permuted. The null models were introduced in Hardy (2008).

Other null model can be used by entering the name of a function. For example, loading the picante package of R, if model=randomizeMatrix, then the permutations will follow function randomizeMatrix available in picante. Any function can be used provided it returns an abundance matrix of similar size as the observed abundance matrix. Parameters of the chosen function can be added to randtest.dpcoa. For example, using parameter null.model of randomizeMatrix, the following command can be used: randtest.dpcoa(xtest,model = randomizeMatrix,null.model = "trialswap")

Value

an object of class randtest

Author(s)

Sandrine Pavoine <pavoine@mnhn.fr>

References


See Also

dpcoa

Examples

data(humDNAm)
dpcoahum <- dpcoa(data.frame(t(humDNAm$samples)), sqrt(humDNAm$distances), scan = FALSE, nf = 2)
randtest(dpcoahum)
Monte-Carlo Test on the percentage of explained (i.e. constrained) inertia

Description

Performs a Monte-Carlo test on the percentage of explained (i.e. constrained) inertia. The statistic is the ratio of the inertia (sum of eigenvalues) of the constrained analysis divided by the inertia of the unconstrained analysis.

Usage

```r
## S3 method for class 'pcaiv'
randtest(xtest, nrepet = 99, ...)
## S3 method for class 'pcaivortho'
randtest(xtest, nrepet = 99, ...)
```

Arguments

- `xtest`: an object of class `pcaiv`, `pcaivortho` or `caiv`;
- `nrepet`: the number of permutations;
- `...`: further arguments passed to or from other methods.

Value

A list of the class `randtest`.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>, original code by Raphaël Pélissier.

Examples

```r
data(rpjdl)
millog <- log(rpjdl$mil + 1)
coa1 <- dudi.coa(rpjdl$fau, scann = FALSE)
caiv1 <- pcaiv(coa1, millog, scan = FALSE)
randtest(caiv1)
```
Description

Functions and classes to manage outputs of two-fold cross-validation for one (class `randxval`) or several (class `krandxval`) statistics

Usage

```r
as.krandxval(RMSEc, RMSEv, quantiles = c(0.25, 0.75), names = colnames(RMSEc), call = match.call())
## S3 method for class 'krandxval'
print(x, ...)
```

```r
as.randxval(RMSEc, RMSEv, quantiles = c(0.25, 0.75), call = match.call())
## S3 method for class 'randxval'
print(x, ...)
```

Arguments

- `RMSEc`: a vector (class `randxval`) or a matrix (class `krandxval`) with the root-mean-square error of calibration (statistics as columns and repetitions as rows)
- `RMSEv`: a vector (class `randxval`) or a matrix (class `krandxval`) with the root-mean-square error of validation (statistics as columns and repetitions as rows)
- `quantiles`: a vector indicating the lower and upper quantiles to compute
- `names`: a vector of names for the statistics
- `call`: the matching call
- `x`: an object of class `randxval` or `krandxval`
- `...`: other arguments to be passed to methods

Value

an object of class `randxval` or `krandxval`

Author(s)

Stéphane Dray (<stephane.dray@univ-lyon1.fr>)

References


See Also

testdim.multiblock
Examples

```r
## an example corresponding to 10 statistics and 100 repetitions
cv <- as.krandxval(RMSEc = matrix(rnorm(1000), nrow = 100), RMSEv =
                   matrix(rnorm(1000, mean = 1), nrow = 100))
cv
if(adegraphicsLoaded())
  plot(cv)
```

### rankrock

**Ordination Table**

**Description**

This data set gives the classification in order of preference of 10 music groups by 51 students.

**Usage**

```r
data(rankrock)
```

**Format**

A data frame with 10 rows and 51 columns.
Each column contains the rank (1 for the favorite, ..., 10 for the less appreciated) attributed to the group by a student.

**Examples**

```r
data(rankrock)
dudi1 <- dudi.pca(rankrock, scannf = FALSE, nf = 3)
if(adegraphicsLoaded()) {
  g <- scatter(dudi1, row.plab.cex = 1.5)
} else {
  scatter(dudi1, clab.r = 1.5)
}
```

### reconst

**Reconstitution of Data from a Duality Diagram**

**Description**

Generic Function for the reconstitution of data from a principal component analysis or a correspondence analysis.
Usage

reconst (dudi, ...)
## S3 method for class 'pca'
reconst(dudi, nf = 1, ...)
## S3 method for class 'coa'
reconst(dudi, nf = 1, ...)

Arguments

dudi an object of class dudi used to select a method: pca or coa
nf an integer indicating the number of kept axes for the reconstitution
... further arguments passed to or from other methods

Value

returns a data frame containing the reconstituted data

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(rhone)
dd1 <- dudi.pca(rhone$tab, nf = 2, scann = FALSE)
rh1 <- reconst(dd1, 1)
rh2 <- reconst(dd1, 2)
par(mfrow = c(4,4))
par(mar = c(2.6,2.6,1.1,1.1))
for (i in 1:15) {
  plot(rhone$date, rhone$tab[,i])
  lines(rhone$date, rh1[,i], lty = 2)
  lines(rhone$date, rh2[,i], lty = 1)
  ade4::scatterutil.sub(names(rhone$tab)[i], 2, "topright")
}
data(chats)
chatsw <- data.frame(t(chats))
chatscoa <- dudi.coa(chatsw, scann = FALSE)
model0 <- reconst(chatscoa, 0)
round(model0,3)
round(chisq.test(chatsw)$expected,3)
chisq.test(chatsw)$statistic
sum(((chatsw-model0)^2)/model0)
effectif <- sum(chatsw)
Genetic structure of two nitrogen fixing bacteria influenced by geographical isolation and host specialization

Description

The data set concerns fixing bacteria belonging to the genus Sinorhizobium (Rhizobiaceae) associated with the plant genus Medicago (Fabaceae). It is a combination of two data sets fully available online from GenBank and published in two recent papers (see reference below). The complete sampling procedure is described in the Additional file 3 of the reference below. We delineated six populations according to geographical origin (France: F, Tunisia Hadjeb: TH, Tunisia Enfidha: TE), the host plant (M. truncatula or similar symbiotic specificity: T, M. lacinia: L), and the taxonomical status of bacteria (S. meliloti: mlt, S. medicae: mdc). Each population will be called hereafter according to the three above criteria, e.g. THLmlt is the population sampled in Tunisia at Hadjeb from M. lacinia nodules which include S. meliloti isolates. S. medicae interacts with M. truncatula while S. meliloti interacts with both M. lacinia (S. meliloti bv. medicaginis) and M. truncatula (S. meliloti bv. meliloti). The numbers of individuals are respectively 46 for FTmdc, 43 for FTMlt, 20 for TETmdc, 24 for TETmlt, 20 for TETmlt, 42 for THTmlt and 20 for THLmlt.

Four different intergenic spacers (IGS), IGSNOD, IGSEXO, IGSGAB, and IGSRPK, distributed on the different replication units of the model strain 1021 of S. meliloti bv. meliloti had been sequenced to characterize each bacterial isolate (DNA extraction and sequencing procedures are described in an additional file). It is noteworthy that the IGSNOD marker is located within the nod gene cluster and that specific alleles at these loci determine the ability of S. meliloti strains to interact with either M. lacinia or M. truncatula.

Usage

data(rhizobium)

Format

rhizobium is a list of 2 components.

- dnaobj: list of dna lists. Each dna list corresponds to a locus. For a given locus, the dna list provides the dna sequences The ith sequences of all loci corresponds to the ith individual of the data set.
- pop: The list of the populations which each individual sequence belongs to.

Source

Examples

```r
# The functions used below require the package ape
data(rhizobium)
if(requireNamespace("ape", quietly = TRUE)) {
  dat <- prep.mdpcoa(rhizobium[[1]], rhizobium[[2]],
        model = c("F84", "F84", "F84", "F81"),
        pairwise.deletion = TRUE)
  sam <- dat$sam
dis <- dat$dis
  # The distances should be Euclidean.
  # Several transformations exist to render a distance object Euclidean
  # (see functions cailliez, lingoes and quasieuclid in the ade4 package).
  # Here we use the quasieuclid function.
  dis <- lapply(dis, quasieuclid)
  mdpcoa1 <- mdpcoa(sam, dis, scann = FALSE, nf = 2)

  # Reference analysis
  plot(mdpcoa1)

  # Differences between the loci
  kplot(mdpcoa1)

  # Alleles projected on the population maps.
  kplotX.mdpcoa(mdpcoa1)
}
```

```r
rhone

Physico-Chemistry Data

Description

This data set gives for 39 water samples a physico-chemical description with the number of sample date and the flows of three tributaries.

Usage

data(rhine)

Format

rhine is a list of 3 components.

- **tab** is a data frame with 39 water samples and 15 physico-chemical variables.
- **date** is a vector of the sample date (in days).
- **disch** is a data frame with 39 water samples and the flows of the three tributaries.
Source


Examples

data(rhone)
pca1 <- dudi.pca(rhone$tab, nf = 2, scann = FALSE)
rh1 <- reconst(pca1, 1)
rh2 <- reconst(pca1, 2)
par(mfrow = c(4,4))
par(mar = c(2.6,2.6,1.1,1.1))
for (i in 1:15) {
  plot(rhone$date, rhone$tab[,i])
  lines(rhone$date, rh1[,i], lwd = 2)
  lines(rhone$date, rh2[,i])
  ade4:::scatterutil.sub(names(rhone$tab)[i], 2, "topright")
}
par(mfrow = c(1,1))

---

**rlq**

*RLQ analysis*

Description

RLQ analysis performs a double inertia analysis of two arrays (R and Q) with a link expressed by a contingency table (L). The rows of L correspond to the rows of R and the columns of L correspond to the rows of Q.

Usage

```
rlq(dudiR, dudiL, dudiQ, scannf = TRUE, nf = 2)
## S3 method for class 'rlq'
print(x, ...)  # S3 method for class 'rlq'
plot(x, xax = 1, yax = 2, ...)  # S3 method for class 'rlq'
summary(object, ...)  # S3 method for class 'rlq'
randtest(xtest,nrepet = 999, modeltype = 6,...)
```

Arguments

- **dudiR**: a duality diagram providing from one of the functions dudi.hillsmith, dudi.pca, ...
- **dudiL**: a duality diagram of the function dudi.coa
dudiQ a duality diagram providing from one of the functions dudi.hillsmith, dudi.pca, ...

scannf a logical value indicating whether the eigenvalues bar plot should be displayed

nf if scannf FALSE, an integer indicating the number of kept axes

x an rlq object

xax the column number for the x-axis

yax the column number for the y-axis

object an rlq object

xtest an rlq object

nrepet the number of permutations

modeltype the model used to permute data(2: permute rows of R, 4: permute rows of Q, 5: permute both, 6: sequential approach, see ter Braak et al. 2012)

... further arguments passed to or from other methods

Value

Returns a list of class 'dudi', sub-class 'rlq' containing:

call call

rank rank

nf a numeric value indicating the number of kept axes

RV a numeric value, the RV coefficient

eig a numeric vector with all the eigenvalues

lw a numeric vector with the rows weights (crossed array)

cw a numeric vector with the columns weigths (crossed array)

tab a crossed array (CA)

li R col = CA row: coordinates

ll R col = CA row: normed scores

ci Q col = CA column: coordinates

cl Q col = CA column: normed scores

lR the row coordinates (R)

mR the normed row scores (R)

lQ the row coordinates (Q)

mQ the normed row scores (Q)

aR the axis onto co-inertia axis (R)

aQ the axis onto co-inertia axis (Q)

WARNING

IMPORTANT: row weights for dudiR and dudiQ must be taken from dudiL.
Note

A testing procedure based on the total coinertia of the RLQ analysis is available by the function `randtest.rlq`. The function allows to deal with various analyses for tables R and Q. Means and variances are recomputed for each permutation (PCA); for MCA, tables are recentred and column weights are recomputed. The case of decentred PCA (PCA where centers are entered by the user) for R or Q is not yet implemented. If you want to use the testing procedure for this case, you must firstly center the table and then perform a non-centered PCA on the modified table.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also

`coinertia, fourthcorner`

Examples

data(aviurba)
coa <- dudi.coa(aviurba$fau, scannf = FALSE, nf = 2)
dudimil <- dudi.hillsmith(aviurba$mil, scannf = FALSE, nf = 2, row.w = coa$lw)
duditrait <- dudi.hillsmith(aviurba$traits, scannf = FALSE, nf = 2, row.w = coa$lw)
rlq1 <- rlq(dudimil, coa, duditrait, scannf = FALSE, nf = 2)
plot(rlq1)
summary(rlq1)
randtest(rlq1)
fourthcorner.rlq(rlq1, type="Q.axes")
fourthcorner.rlq(rlq1, type="R.axes")
Description

This data set gives the abundance of 51 species and 8 environmental variables in 182 sites.

Usage

data(rpjdl)

Format

rpjdl is a list of 5 components.

fau is the faunistic array of 182 sites (rows) and 51 species (columns).

mil is the array of environmental variables: 182 sites and 8 variables.

frlab is a vector of the names of species in French.

lalab is a vector of the names of species in Latin.

lab is a vector of the simplified labels of species.

Source


References


Examples

## Not run:
data(rpjdl)
coa1 <- dudi.coa(rpjdl$fau, scann = FALSE)
pca1 <- dudi.pca(rpjdl$fau, scal = FALSE, scann = FALSE)
if(adegraphicsLoaded()) {
  g1 <- s.distri(coa1$l1, rpjdl$fau, xax = 2, yax = 1, starSize = 0.3, ellipseSize = 0, plab.cex = 0)
  g2 <- s.distri(pca1$l1, rpjdl$fau, xax = 2, yax = 1, starSize = 0.3, ellipseSize = 0, plab.cex = 0)
} else {
rtest

Class of the Permutation Tests (in R).

Description

rtest is a generic function. It proposes methods for the following objects between, discrimin, procuste...

Usage

rtest(xtest, ...)

Arguments

xtest an object used to select a method
...

Value

rtest returns an object of class randtest

Author(s)

Daniel Chessel

See Also

RV.rtest, mantel.rtest, procuste.rtest, randtest

Examples

par(mfrow = c(2, 2))
for (x0 in c(2.4, 3.4, 5.4, 20.4)) {
  l0 <- as.randtest(sim = rnorm(200), obs = x0)
  print(l0)
  plot(l0, main = paste("p.value = ", round(l0$pvalue, dig = 5)))
}
par(mfrow = c(1, 1))
rtest.between

Monte-Carlo Test on the between-groups inertia percentage (in R).

Description

Performs a Monte-Carlo test on the between-groups inertia percentage.

Usage

## S3 method for class 'between'

rtest(xtest, nrepet = 99, ...)

Arguments

- `xtest`: an object of class between
- `nrepet`: the number of permutations
- `...`: further arguments passed to or from other methods

Value

a list of the class rtest

Author(s)

Daniel Chessel

References


Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 3)
rand1 <- rtest(bca(pca1, meaudret$design$season, scan = FALSE), 99)
rand1
plot(rand1, main = "Monte-Carlo test")
rtest.discrimin  
**Monte-Carlo Test on a Discriminant Analysis (in R).**

**Description**

Test of the sum of a discriminant analysis eigenvalues (divided by the rank). Non parametric version of the Pillai’s test. It authorizes any weighting.

**Usage**

```r
## S3 method for class 'discrimin'
rtest(xtest, nrepet = 99, ...)
```

**Arguments**

- `xtest`: an object of class `discrimin`
- `nrepet`: the number of permutations
- `...`: further arguments passed to or from other methods

**Value**

returns a list of class `rtest`

**Author(s)**

Daniel Chessel

**Examples**

```r
data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 3)
rand1 <- rtest(discrimin(pca1, meaudret$design$season, scan = FALSE), 99)
rand1
```

#Monte-Carlo test
#Observation: 0.3035
#Call: as.rtest(sim = sim, obs = obs)
#Based on 999 replicates
#Simulated p-value: 0.001

```r
plot(rand1, main = "Monte-Carlo test")
```

```r
summary.manova(manova(as.matrix(meaudret$env)~meaudret$design$season), "Pillai")
```

```
#  Df Pillai approx F num Df den Df Pr(>F)
# meaudret$design$season 3 2.731 11.300 27 30 1.6e-09 ***
# Residuals 16
```

# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
# 2.731/9 = 0.3034
RV.rtest  

Monte-Carlo Test on the sum of eigenvalues of a co-inertia analysis (in R).

Description

performs a Monte-Carlo Test on the sum of eigenvalues of a co-inertia analysis.

Usage

RV.rtest(df1, df2, nrepet = 99, ...)

Arguments

df1, df2  two data frames with the same rows
nrepet    the number of permutations
...      further arguments passed to or from other methods

Value

returns a list of class 'rtest'

Author(s)

Daniel Chessel

References


Examples

data(doubs)
pca1 <- dudi.pca(doubs$env, scal = TRUE, scann = FALSE)
pca2 <- dudi.pca(doubs$fish, scal = FALSE, scann = FALSE)
rv1 <- RV.rtest(pca1$tab, pca2$tab, 99)
rv1
plot(rv1)
RVdist.randtest

Tests of randomization on the correlation between two distance matrices (in R).

Description

performs a RV Test between two distance matrices.

Usage

RVdist.randtest(m1, m2, nrepet = 999, ...)

Arguments

m1, m2 two Euclidean matrices
nrepet the number of permutations
... further arguments passed to or from other methods

Value

returns a list of class 'randtest'

Author(s)

Daniel Chessel

References


s.arrow

Plot of the factorial maps for the projection of a vector basis

Description

performs the scatter diagrams of the projection of a vector basis.

Usage

s.arrow(dfxy, xax = 1, yax = 2, label = row.names(dfxy),
clabel = 1, pch = 20, cpoint = 0, boxes = TRUE, edge = TRUE, origin = c(0,0),
xlim = NULL, ylim = NULL, grid = TRUE, addaxes = TRUE, cgrid = 1,
sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL,
contour = NULL, area = NULL, add.plot = FALSE)
**Arguments**

- `dfxy`: a data frame containing the two columns for the axes.
- `xax`: the column number of x in `dfxy`.
- `yax`: the column number of y in `dfxy`.
- `label`: a vector of strings of characters for the point labels.
- `clabel`: if not NULL, a character size for the labels used with `par("cex")*clabel`.
- `pch`: if `cpoint` > 0, an integer specifying the symbol or the single character to be used in plotting points.
- `cpoint`: a character size for plotting the points, used with `par("cex")*cpoint`. If zero, no points are drawn.
- `boxes`: if TRUE, labels are framed.
- `edge`: a logical value indicating whether the arrows should be plotted.
- `origin`: the fixed point in the graph space, by default `c(0,0)` the origin of axes. The arrows begin at `cent`.
- `xlim`: the ranges to be encompassed by the x-axis, if NULL they are computed.
- `ylim`: the ranges to be encompassed by the y-axis, if NULL they are computed.
- `grid`: a logical value indicating whether a grid in the background of the plot should be drawn.
- `addaxes`: a logical value indicating whether the axes should be plotted.
- `cgrid`: a character size, parameter used with `par("cex")*cgrid`, to indicate the mesh of the grid.
- `sub`: a string of characters to be inserted as legend.
- `csub`: a character size for the legend, used with `par("cex")*csub`.
- `posssub`: a string of characters indicating the legend position ("topleft", "topright", "bottomleft", "bottomright").
- `pixmap`: an object 'pixmap' displayed in the map background.
- `contour`: a data frame with 4 columns to plot the contour of the map: each row gives a segment (x1,y1,x2,y2).
- `area`: a data frame of class 'area' to plot a set of surface units in contour.
- `add.plot`: if TRUE uses the current graphics window.

**Value**

The matched call.

**Author(s)**

Daniel Chessel

**Examples**

```r
s.arrow(cbind.data.frame(runif(55,-2,3), runif(55,-3,2)))
```
Plot of the factorial maps with polygons of contour by level of a factor

Description

performs the scatter diagrams with polygons of contour by level of a factor.

Usage

`s.chull(dfxy, fac, xax = 1, yax = 2,
optchull = c(0.25, 0.5, 0.75, 1), label = levels(fac), clabel = 1,
cpoint = 0, col = rep(1, length(levels(fac))), xlim = NULL, ylim = NULL,
grid = TRUE, addaxes = TRUE, origin = c(0,0), include.origin = TRUE,
sub = "", csub = 1, possub = "bottomleft", cgrid = 1, pixmap = NULL,
contour = NULL, area = NULL, add.plot = FALSE)`

Arguments

dfxy a data frame containing the two columns for the axes
fac a factor partitioning the rows of the data frame in classes
xax the column number of x in dfxy
yax the column number of y in dfxy
optchull the number of convex hulls and their interval
label a vector of strings of characters for the point labels
clabel if not NULL, a character size for the labels, used with par("cex")*clabel
cpoint a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
col a vector of colors used to draw each class in a different color
xlim the ranges to be encompassed by the x axis, if NULL, they are computed
ylim the ranges to be encompassed by the y axis, if NULL they are computed
grid a logical value indicating whether a grid in the background of the plot should be drawn
addaxes a logical value indicating whether the axes should be plotted
origin the fixed point in the graph space, for example c(0,0) the origin axes
include.origin a logical value indicating whether the point "origin" should be belonged to the graph space
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
**s.class**

Plot of factorial maps with representation of point classes

**Description**

performs the scatter diagrams with representation of point classes.

**Usage**

```r
s.class(dfxy, fac, wt = rep(1, length(fac)), xax = 1, yax = 2, cstar = 1, cellipse = 1.5, axesell = TRUE, label = levels(fac), clabel = 1, cpoint = 1, pch = 20, col = rep(1, length(levels(fac))), xlim = NULL, ylim = NULL, grid = TRUE, addaxes = TRUE, origin = c(0,0), include.origin = TRUE, sub = "", csub = 1, possub = "bottomleft", cgrid = 1, pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)
```
Arguments

- **dfxy**: a data frame containing the two columns for the axes
- **fac**: a factor partitioning the rows of the data frame in classes
- **wt**: a vector of the point weightings of the data frame used for computing the means (star centers) and the ellipses of dispersion
- **xax**: the column number of x in dfxy
- **yax**: the column number of y in dfxy
- **cstar**: a number between 0 and 1 which defines the length of the star size
- **cellipse**: a positive coefficient for the inertia ellipse size
- **axesell**: a logical value indicating whether the ellipse axes should be drawn
- **label**: a vector of strings of characters for the point labels
- **clabel**: if not NULL, a character size for the labels, used with `par("cex")`*clabel
- **cpoint**: a character size for plotting the points, used with `par("cex")`*cpoint. If zero, no points are drawn
- **pch**: if cpoint > 0, an integer specifying the symbol or the single character to be used in plotting points
- **col**: a vector of colors used to draw each class in a different color
- **xlim**: the ranges to be encompassed by the x, if NULL they are computed
- **ylim**: the ranges to be encompassed by the y, if NULL they are computed
- **grid**: a logical value indicating whether a grid in the background of the plot should be drawn
- **addaxes**: a logical value indicating whether the axes should be plotted
- **origin**: the fixed point in the graph space, for example c(0,0) the origin axes
- **include.origin**: a logical value indicating whether the point "origin" should be belonged to the graph space
- **sub**: a string of characters to be inserted as legend
- **csub**: a character size for the legend, used with `par("cex")`*csub
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- **cgrid**: a character size, parameter used with `par("cex")`* cgrid to indicate the mesh of the grid
- **pixmap**: an object 'pixmap' displayed in the map background
- **contour**: a data frame with 4 columns to plot the contour of the map : each row gives a segment (x1,y1,x2,y2)
- **area**: a data frame of class 'area' to plot a set of surface units in contour
- **add.plot**: if TRUE uses the current graphics window

Value

The matched call.
### Examples

```r
if(!adegraphicsLoaded()) {
  xy <- cbind.data.frame(x = runif(200, -1, 1), y = runif(200, -1, 1))
  posi <- factor(xy$x > 0) : factor(xy$y > 0)
  coul <- c("black", "red", "green", "blue")
  par(mfrow = c(2, 2))
  s.class(xy, posi, cpoi = 2)
  s.class(xy, posi, cell = 0, cstar = 0.5)
  s.class(xy, posi, cell = 2, axesell = FALSE, csta = 0, col = coul)
  s.chull(xy, posi, cpoi = 1)
  par(mfrow = c(1, 1))
}
```

```r
## Not run:
data(banque)
dudi1 <- dudi.acm(banque, scannf = FALSE)
coul = rainbow(length(levels(banque[, 20])))
par(mfrow = c(2, 2))
s.label(dudi1$li, sub = "Factorial map from ACM", csub = 1.5,
       possub = "topleft")
s.class(dudi1$li, banque[, 20], sub = names(banque)[20],
        possub = "bottomright", cell = 0, cstar = 0.5, cgrid = 0, csub = 1.5)
s.class(dudi1$li, banque[, 20], csta = 0, cell = 2, cgrid = 0,
        clab = 1.5)
s.class(dudi1$li, banque[, 20], sub = names(banque)[20],
        possub = "topright", cgrid = 0, col = coul)
par(mfrow = c(1, 1))
```

```r
par(mfrow = n2mfrow(ncol(banque)))
for(i in 1:ncol(banque))
  s.class(dudi1$li, banque[, i], clab = 1.5, sub = names(banque)[i],
          csub = 2, possub = "topleft", cgrid = 0, csta = 0, cpoi = 0)
s.label(dudi1$li, clab = 0, sub = "Common background")
par(mfrow = c(1, 1))
```

## End(Not run)
```

### Description

`s.corcircle` performs the scatter diagram of a correlation circle.
Usage

```r
s.corcircle(dfxy, xax = 1, yax = 2, label = row.names(df),
            clabel = 1, grid = TRUE, sub = "", csub = 1, possub = "bottomleft",
            cgrid = 0, fullcircle = TRUE, box = FALSE, add.plot = FALSE)
```

Arguments

dfxy a data frame with two coordinates
xax the column number for the x-axis
yax the column number for the y-axis
label a vector of strings of characters for the point labels
clabel if not NULL, a character size for the labels, used with par("cex")*clabel
grid a logical value indicating whether a grid in the background of the plot should be drawn
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
fullcircle a logical value indicating whether the complete circle should be drawn
box a logical value indicating whether a box should be drawn
add.plot if TRUE uses the current graphics window

Value

The matched call.

Author(s)

Daniel Chessel

Examples

```r
if(!adegraphicsLoaded()) {
  data (olympic)
  dudi1 <- dudi.pca(olympic$tab, scan = FALSE) # a normed PCA
  par(mfrow = c(2, 2))
  s.corcircle(dudi1$co, lab = names(olympic$tab))
  s.corcircle(dudi1$co, cgrid = 0, full = FALSE, clab = 0.8)
  s.corcircle(dudi1$co, lab = as.character(1:11), cgrid = 2,
              full = FALSE, sub = "Correlation circle", csub = 2.5,
              possub = "bottomleft", box = TRUE)
  s.arrow(dudi1$co, clab = 1)
  par(mfrow = c(1, 1))
}
```
**Plot of a frequency distribution**

**Description**

performs the scatter diagram of a frequency distribution.

**Usage**

```r
s.distri(dfxy, dfdistri, xax = 1, yax = 2, cstar = 1,
cellipse = 1.5, axesell = TRUE, label = names(dfdistri),
clabel = 0, cpoint = 1, pch = 20, xlim = NULL, ylim = NULL,
grid = TRUE, addaxes = TRUE, origin = c(0,0),
include.origin = TRUE, sub = "", csub = 1, possub = "bottomleft",
cgrid = 1, pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)
```

**Arguments**

- **dfxy**: a data frame containing two columns for the axes
- **dfdistr**: a data frame containing the mass distributions in columns
- **xax**: the column number for the x-axis
- **yax**: the column number for the y-axis
- **cstar**: a number between 0 and 1 which defines the length of the star size
- **cellipse**: a positive coefficient for the inertia ellipse size
- **axesell**: a logical value indicating whether the ellipse axes should be drawn
- **label**: a vector of strings of characters for the distribution centers labels
- **clabel**: if not NULL, a character size for the labels, used with `par("cex")*clabel`
- **cpoint**: a character size for plotting the points, used with `par("cex")*cpoint`. If zero, no points are drawn
- **pch**: if `cpoint` > 0, an integer specifying the symbol or the single character to be used in plotting points
- **xlim**: the ranges to be encompassed by the x, if NULL they are computed
- **ylim**: the ranges to be encompassed by the y, if NULL they are computed
- **grid**: a logical value indicating whether a grid in the background of the plot should be drawn
- **addaxes**: a logical value indicating whether the axes should be plotted
- **origin**: the fixed point in the graph space, for example `c(0,0)` the origin axes
- **include.origin**: a logical value indicating whether the point "origin" should be belonged to the graph space
- **sub**: a string of characters to be inserted as legend
- **csub**: a character size for the legend, used with `par("cex")*csub`
possub  a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

cgrid  a character size, parameter used with par("cex")* cgrid to indicate the mesh of the grid

pixmap  an object ‘pixmap’ displayed in the map background

contour  a data frame with 4 columns to plot the contour of the map: each row gives a segment (x1,y1,x2,y2)

area  a data frame of class ‘area’ to plot a set of surface units in contour

add.plot  if TRUE uses the current graphics window

Value

The matched call.

Author(s)

Daniel Chessel

Examples

if(!adegraphicsLoaded()) {
  xy <- cbind.data.frame(x = runif(200, -1, 1), y = runif(200, -1, 1))
  distri <- data.frame(w1 = rpois(200, xy$x * (xy$x > 0)))
  s.value(xy, distri$w1, cpoi = 1)
  s.distri(xy, distri, add.p = TRUE)

  w1 <- as.numeric((xy$x > 0) & (xy$y > 0))
  w2 <- ((xy$x > 0) & (xy$y < 0)) * (1 - xy$y) * xy$x
  w3 <- ((xy$x < 0) & (xy$y > 0)) * (1 - xy$x) * xy$y
  w4 <- ((xy$x < 0) & (xy$y < 0)) * xy$y * xy$x

  distri <- data.frame(a = w1 / sum(w1), b = w2 / sum(w2),
                       c = w3 / sum(w3), d = w4 / sum(w4))
  s.value(xy, unlist(apply(distri, 1, sum)), cleg = 0, csi = 0.75)
  s.distri(xy, distri, clab = 2, add.p = TRUE)

  data(rpjdl)
  xy <- dudi.coa(rpjdl$fau, scan = FALSE)$li
  par(mfrow = c(3, 4))
  for (i in c(1, 5, 8, 20, 21, 23, 26, 33, 36, 44, 47, 49)) {
    s.distri(xy, rpjdl$fau[, i], cell = 1.5, sub = rpjdl$frlab[i],
              csub = 2, cgrid = 1.5))
    par(mfrow = c(1, 1))
  }
}
**s.hist**

*Display of a scatterplot and its two marginal histograms*

**Description**
performs a scatterplot and the two marginal histograms of each axis.

**Usage**
s.hist(dfxy, xax = 1, yax = 2, cgrid = 1, cbreaks = 2, adjust = 1, ...)

**Arguments**
- **dfxy**: a data frame with two coordinates
- **xax**: column for the x axis
- **yax**: column for the y axis
- **cgrid**: a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
- **cbreaks**: a parameter used to define the numbers of cells for the histograms. By default, two cells are defined for each interval of the grid displayed in s.label. With an increase of the integer cbreaks, the number of cells increases as well.
- **adjust**: a parameter passed to density to display a kernel density estimation
- **...**: further arguments passed from the s.label for the scatter plot

**Value**
The matched call.

**Author(s)**
Daniel Chessel

**Examples**
```r
data(rpjdl)
coa1 <- dudi.coa(rpjdl$fau, scannf = FALSE, nf = 4)
s.hist(coa1$li)
s.hist(coa1$li, cgrid = 2, cbr = 3, adj = 0.5, clab = 0)
s.hist(coa1$co, cgrid = 2, cbr = 3, adj = 0.5, clab = 0)
```
**s.image**

*Graph of a variable using image and contour*

**Description**

performs a scatterplot

**Usage**

```r
s.image(dfxy, z, xax = 1, yax = 2, span = 0.5, xlim = NULL, ylim = NULL, kgrid = 2, scale = TRUE, grid = FALSE, addaxes = FALSE, cgrid = 0, include.origin = FALSE, origin = c(0, 0), sub = "", csub = 1, possub = "topleft", neig = NULL, cneig = 1, image.plot = TRUE, contour.plot = TRUE, pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)
```

**Arguments**

- `dfxy`: a data frame containing the two columns for the axes
- `z`: a vector of values on the dfxy rows
- `xax`: the column number of x in dfxy
- `yax`: the column number of y in dfxy
- `span`: the parameter alpha which controls the degree of smoothing
- `xlim`: the ranges to be encompassed by the x-axis, if NULL they are computed
- `ylim`: the ranges to be encompassed by the y-axis, if NULL they are computed
- `kgrid`: a number of points used to locally estimate the level line through the nodes of the grid, used by kgrid*sqrt(length(z))
- `scale`: if TRUE, data are centered and reduced
- `grid`: if TRUE, the background grid is traced
- `addaxes`: a logical value indicating whether the axes should be plotted
- `cgrid`: a character size, parameter used with par("cex")* cgrid to indicate the mesh of the grid
- `include.origin`: a logical value indicating whether the point "origin" should be belonged to the graph space
- `origin`: the fixed point in the graph space, for example c(0,0) the origin axes
- `sub`: a string of characters to be inserted as legend
- `csub`: a character size for the legend, used with par("cex")*csub
- `possub`: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- `neig`: an object of class neig
- `cneig`: a size for the neighbouring graph lines used with par("lwd")*cneig
image.plot  if TRUE, the image is traced
contour.plot if TRUE, the contour lines are plotted
pixmap       an object 'pixmap' displayed in the map background
contour     a data frame with 4 columns to plot the contour of the map : each row gives a segment (x1,y1,x2,y2)
area         a data frame of class 'area' to plot a set of surface units in contour
add.plot     if TRUE uses the current graphics window

Value

The matched call.

Author(s)

Daniel Chessel

Examples

if(!adegraphicsLoaded()) {
  if(requireNamespace("splancs", quietly = TRUE)) {
    wxy <- data.frame(expand.grid(-3:3, -3:3))
    names(wxy) <- c("x", "y")
    z <- (1 / sqrt(2)) * exp(-((wxy$x ^ 2 + wxy$y ^ 2) / 2)
    par(mfrow = c(2, 2))
    s.value(wxy, z)
    s.image(wxy, z)
    s.image(wxy, z, kgrid = 5)
    s.image(wxy, z, kgrid = 15)
    par(mfrow = c(1, 1))
  }
  ## Not run:
  data(t3012)
  if(requireNamespace("splancs", quietly = TRUE)) {
    par(mfrow = c(3, 4))
    for(k in 1:12)
      s.image(t3012$xy, scalewt(t3012$temp[, k]), kgrid = 3)
    par(mfrow = c(1, 1))
  }
  data(elec88)
  if(requireNamespace("splancs", quietly = TRUE)) {
    par(mfrow = c(3,4))
    for(k in 1:12)
      s.image(t3012$xy, scalewt(t3012$temp[, k], kgrid = 3, sub = names(t3012/temp)[k],
                 csub = 3, area = elec88$area)
    par(mfrow = c(1, 1))
  }
  ## End(Not run)
}
**s.kde2d**

*Scatter Plot with Kernel Density Estimate*

**Description**

performs a scatter of points without labels by a kernel Density Estimation in One or Two Dimensions

**Usage**

`s.kde2d(dfxy, xax = 1, yax = 2, pch = 20, cpoint = 1, neig = NULL, cneig = 2, xlim = NULL, ylim = NULL, grid = TRUE, addaxes = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0, 0), sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)`

**Arguments**

- **dfxy**: a data frame with at least two coordinates
- **xax**: the column number for the x-axis
- **yax**: the column number for the y-axis
- **pch**: if `cpoint > 0`, an integer specifying the symbol or the single character to be used in plotting points
- **cpoint**: a character size for plotting the points, used with `par("cex")*cpoint`. If zero, no points are drawn
- **neig**: a neighbouring graph
- **cneig**: a size for the neighbouring graph lines used with `par("lwd")*cneig`
- **xlim**: the ranges to be encompassed by the x axis, if NULL, they are computed
- **ylim**: the ranges to be encompassed by the y axis, if NULL, they are computed
- **grid**: a logical value indicating whether a grid in the background of the plot should be drawn
- **addaxes**: a logical value indicating whether the axes should be plotted
- **cgrid**: a character size, parameter used with `par("cex")* 'cgrid'` to indicate the mesh of the grid
- **include.origin**: a logical value indicating whether the point "origin" should be belonged to the graph space
- **origin**: the fixed point in the graph space, for example c(0,0) the origin axes
- **sub**: a string of characters to be inserted as legend
- **csub**: a character size for the legend, used with `par("cex")*csub`
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- **pixmap**: an object pixmap displayed in the map background
s.label

297

contour

a data frame with 4 columns to plot the contour of the map : each row gives a
segment (x1,y1,x2,y2)

area

a data frame of class ’area’ to plot a set of surface units in contour

add.plot

if TRUE uses the current graphics window

Value
The matched call.
Author(s)
Daniel Chessel
Examples
# To recognize groups of points
if(!adegraphicsLoaded()) {
data(rpjdl)
coa1 <- dudi.coa(rpjdl$fau, scannf = FALSE, nf = 3)
s.kde2d(coa1$li)
}

s.label

Scatter Plot

Description
performs the scatter diagrams with labels.
Usage
s.label(dfxy, xax = 1, yax = 2, label = row.names(dfxy),
clabel = 1, pch = 20, cpoint = if (clabel == 0) 1 else 0, boxes = TRUE,
neig = NULL, cneig = 2, xlim = NULL, ylim = NULL, grid = TRUE,
addaxes = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0,0),
sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL,
contour = NULL, area = NULL, add.plot = FALSE)
Arguments
dfxy

a data frame with at least two coordinates

xax

the column number for the x-axis

yax

the column number for the y-axis

label

a vector of strings of characters for the point labels

clabel

if not NULL, a character size for the labels, used with par("cex")*clabel

pch

if cpoint > 0, an integer specifying the symbol or the single character to be used
in plotting points


cpoint  a character size for plotting the points, used with \texttt{par("cex")}\*cpoint. If zero, no points are drawn

boxes  if \texttt{TRUE}, labels are framed

neig  a neighbouring graph

cneig  a size for the neighbouring graph lines used with \texttt{par("lwd")}\*cneig

xlim  the ranges to be encompassed by the \texttt{x} axis, if \texttt{NULL}, they are computed

ylim  the ranges to be encompassed by the \texttt{y} axis, if \texttt{NULL}, they are computed

grid  a logical value indicating whether a grid in the background of the plot should be drawn

addaxes  a logical value indicating whether the axes should be plotted

cgrid  a character size, parameter used with \texttt{par("cex")}\*cgrid to indicate the mesh of the grid

include.origin  a logical value indicating whether the point "origin" should be belonged to the graph space

origin  the fixed point in the graph space, for example \texttt{c(0,0)} the origin axes

sub  a string of characters to be inserted as legend

csub  a character size for the legend, used with \texttt{par("cex")}\*csub

possub  a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

pixmap  an object 'pixmap' displayed in the map background

contour  a data frame with 4 columns to plot the contour of the map: each row gives a segment \texttt{(x1,y1,x2,y2)}

area  a data frame of class 'area' to plot a set of surface units in contour

add.plot  if \texttt{TRUE} uses the current graphics window

\section*{Value}

The matched call.

\section*{Author(s)}

Daniel Chessel

\section*{Examples}

\begin{verbatim}
if(!adeographicsLoaded()) {
    layout(matrix(c(1, 2, 3, 2), 2, 2))
data(atlas)
s.label(atlas$xy, lab = atlas$names.district,
    area = atlas$area, inc = FALSE, addax = FALSE)
data(mafragh)
s.label(mafragh$xy, inc = FALSE, neig = mafragh$neig, addax = FALSE)
data(irishdata)
s.label(irishdata$xy, inc = FALSE, contour = irishdata$contour,
    addax = FALSE)
\end{verbatim}
```r
par(mfrow = c(2, 2))
cha <- ls()
s.label(cbind.data.frame(runif(length(cha)), runif(length(cha))), lab = cha)
x <- runif(50, -2, 2)
y <- runif(50, -2, 2)
z <- x^2 + y^2
s.label(data.frame(x, y), lab = as.character(z < 1))
s.label(data.frame(x, y), clab = 0, cpoi = 1, add.plot = TRUE)
symbols(0, 0, circles = 1, add = TRUE, inch = FALSE)
s.label(cbind.data.frame(runif(100, 0, 10), runif(100, 5, 12)), incl = FALSE, clab = 0)
s.label(cbind.data.frame(runif(100, -3, 12), runif(100, 2, 10)), cl = 0, cp = 2, include = FALSE)
```
ylim the ranges to be encompassed by the y axis, if NULL, they are computed
grid a logical value indicating whether a grid in the background of the plot should be drawn
addaxes a logical value indicating whether the axes should be plotted
cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
include.origin a logical value indicating whether the point "origin" should be belonged to the graph space
origin the fixed point in the graph space, for example c(0,0) the origin axes
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
pixmap an object ‘pixmap’ displayed in the map background
contour a data frame with 4 columns to plot the contour of the map: each row gives a segment (x1,y1,x2,y2)
area a data frame of class ‘area’ to plot a set of surface units in contour
add.plot if TRUE uses the current graphics window

Value
The matched call.

Author(s)
Daniel Chessel and Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples
if(requireNamespace("pixmap", quietly = TRUE) & requireNamespace("sp", quietly = TRUE)) {
  if(!adegraphicsLoaded()) {
    data(ggtortoises)
a1 <- ggtortoises$area
area.plot(a1)
rect(min(a1$x), min(a1$y), max(a1$x), max(a1$y), col = "lightblue")
invisible(lapply(split(a1, a1$id), function(x) polygon(x[, -1],col = "white")))
s.label(ggtortoises$misc, grid = FALSE, include.ori = FALSE, addaxes = FALSE, add.p = TRUE)
listico <- ggtortoises$ico[as.character(ggtortoises$pop$carap)]
s.logo(ggtortoises$pop, listico, add.p = TRUE)
  } else {
    data(capitales, package = "ade4")
    # 'capitales' data doesn’t work with ade4 anymore
g3 <- s.logo(capitales$xy[sort(rownames(capitales$xy))], capitales$logo,
      Sp = capitales$Spatial, pbackground.col = "lightblue", pSp.col = "white",
pgrid.draw = FALSE)
  }
}


s.match

Plot of Paired Coordinates

Description

performs the scatter diagram for a paired coordinates.

Usage

s.match(df1xy, df2xy, xax = 1, yax = 2, pch = 20, cpoint = 1,
        label = row.names(df1xy), clabel=1, edge = TRUE, xlim = NULL,
        ylim = NULL, grid = TRUE, addaxes = TRUE, cgrid = 1,
        include.origin = TRUE, origin = c(0,0), sub = "", csub = 1.25,
        possub = "bottomleft", pixmap = NULL, contour = NULL, area = NULL,
        add.plot = FALSE)

Arguments

df1xy a data frame containing two columns from the first system
df2xy a data frame containing two columns from the second system
xax the column number for the x-axis of both the two systems
yax the column number for the y-axis of both the two systems
pch if cpoint > 0, an integer specifying the symbol or the single character to be used in plotting points
    cpoint a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
    label a vector of strings of characters for the couple labels
clabel if not NULL, a character size for the labels, used with par("cex")*clabel
edge If TRUE the arrows are plotted, otherwise only the segments are drawn
xlim the ranges to be encompassed by the x axis, if NULL they are computed
ylim the ranges to be encompassed by the y axis, if NULL they are computed
grid a logical value indicating whether a grid in the background of the plot should be drawn
addaxes a logical value indicating whether the axes should be plotted
cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
include.origin a logical value indicating whether the point "origin" should be belonged to the graph space
origin the fixed point in the graph space, for example c(0,0) the origin axes
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
s.match.class

**Description**
Performs a graphical representation of two sets of coordinates (different colors and symbols) and a partitionning into classes

**Usage**
```r
s.match.class(df1xy, df2xy, fac, wt = rep(1/nrow(df1xy), nrow(df1xy)),
   xax = 1, yax = 2, pch1 = 16, pch2 = 15, col1 = rep("lightgrey",
   nlevels(fac)), col2 = rep("darkgrey", nlevels(fac)), cpoint = 1, label =
   levels(fac), clabel = 1, cstar = 1, cellipse = 0, axesell = TRUE, xlim =
   NULL, ylim = NULL, grid = TRUE, addaxes = TRUE, cgrid = 1,
   include.origin = TRUE, origin = c(0, 0), sub = "", csub = 1.25, possub =
   "bottomleft", pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)
```

**Value**
The matched call.

**Author(s)**
Daniel Chessel

**Examples**
```r
if(!adegraphicsLoaded()) {
   X <- data.frame(x = runif(50, -1, 2), y = runif(50, -1, 2))
   Y <- X + rnorm(100, sd = 0.3)
   par(mfrow = c(2, 2))
   s.match(X, Y)
   s.match(X, Y, edge = FALSE, clab = 0)
   s.label(X, clab = 1, add.plot = TRUE)
   s.label(Y, clab = 0.75, add.plot = TRUE)
   s.match(Y, X, clab = 0)
   par(mfrow = c(1, 1))
}
```
Arguments

df1xy  a dataframe with the first system of coordinates

df2xy  a dataframe with the second system of coordinates

fac    a factor partitioning the rows of the data frame in classes

wt     a vector of weights

xax    a number indicating which column should be plotted on the x-axis

yax    a number indicating which column should be plotted on the x-axis

pch1   if cpoint > 0, an integer specifying the symbol or the single character to be used for plotting points

pch2   if cpoint > 0, an integer specifying the symbol or the single character to be used for plotting points

col1   a color for symbols

col2   a color for symbols

cpoint a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn

label  a vector of strings of characters for the couple labels

clabel if not NULL, a character size for the labels, used with par("cex")*clabel

cstar  a number between 0 and 1 which defines the length of the star size

cellipse a positive coefficient for the inertia ellipse size

axesell a logical value indicating whether the ellipse axes should be drawn

xlim   the ranges to be encompassed by the x axis, if NULL they are computed

ylim   the ranges to be encompassed by the y axis, if NULL they are computed

grid   a logical value indicating whether a grid in the background of the plot should be drawn

addaxes a logical value indicating whether the axes should be plotted

cgrid  a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid

include.origin a logical value indicating whether the point "origin" should belong to the graph space

origin  a fixed point in the graph space, for example c(0,0) for the origin of axes

sub     a string of characters to be inserted as legend

csub    a character size for the legend, used with par("cex")*csub

possub  a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
pixmap  a pixmap object

contour a dataframe with 4 columns to plot the contour of the map : each row gives a segment (x1,y1,x2,y2)

area    a dataframe of class 'area' to plot an areal map

add.plot if TRUE, add the plot to the current graphic device
Value

The matched call.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

See Also

s.class, s.match

Examples

```r
xy <- data.frame(matrix(rnorm(100), 50, 2))
xy[, 1] <- xy[, 1] + rep(seq(0, 12, by = 3), rep(10, 5))
xy[, 2] <- xy[, 2] + rep(seq(0, 12, by = 3), rep(10, 5))
fac <- gl(5, 10)
xy2 <- xy + matrix(rnorm(100), 50, 2) + 1

if(adeGraphicsLoaded()) {
  mat <- rbind(xy, xy2)
  minmat <- apply(mat, 2, min)
  maxmat <- apply(mat, 2, max)
  lag <- 0.1 * abs(minmat - maxmat)
  xli <- c(minmat[1] - lag[1], maxmat[1] + lag[1])

  g1 <- s.class(xy, fac, ellipseSize = 0, col = rep("grey45", nlevels(fac)), xlim = xli, ylim = yli, plabels.cex = 0, plot = FALSE)
  g2 <- s.class(xy2, fac, ellipseSize = 0, col = rep("grey75", nlevels(fac)), xlim = xli, ylim = yli, plabels.cex = 0, plot = FALSE)
  g3 <- s.match(g1@stats$means, g2@stats$means, xlim = xli, ylim = yli, plines.lwd = 2, psub.text = "xy -> xy2", plot = FALSE)

  g4 <- do.call("superpose", list(g1, g2))
  g4@Call <- call("superpose", g1@Call, g2@Call)
  g4 <- do.call("superpose", list(g4, g3))
  g4@Call <- call("superpose", g4@Call, g3@Call)
  g4
}
else {
  s.match.class(xy, xy2, fac)
}
```

---

**s.multinom**

Graph of frequency profiles (useful for instance in genetic)
The main purpose of this function is to draw categories using scores and profiles by their gravity center. Confidence intervals of the average position (issued from a multinomial distribution) can be superimposed.

Usage

```r
s.multinom(dfxy, dfrowprof, translate = FALSE, xax = 1, yax = 2,
labelcat = row.names(dfxy), clabelcat = 1, cpointcat = if (clabelcat == 0) 2 else 0,
labelrowprof = row.names(dfrowprof), clabelrowprof = 0.75,
cpointrowprof = if (clabelrowprof == 0) 2 else 0, pchrowprof = 20,
coulrowprof = grey(0.8), proba = 0.95, n.sample = apply(dfrowprof, 1, sum),
axesell = TRUE, ...)
```

Arguments

dfxy is a data frame containing at least two numerical variables. The rows of dfxy are categories such as 1,2 and 3 in the triangular plot.

dfrowprof is a data frame whose the columns are the rows of dfxy. The rows of dfxy are profiles or frequency distributions on the categories. The column number of dfrowprof must be equal to the row number of dfxy. row.names(dfxy) and names(dfrowprof) must be identical.

translate a logical value indicating whether the plot should be translated(TRUE) or not. The origin becomes the gravity center weighted by profiles.

xax the column number of dfxy for the x-axis

yax the column number of dfxy for the y-axis

labelcat a vector of strings of characters for the labels of categories

clabelcat an integer specifying the character size for the labels of categories, used with par("cex")*clabelcat

cpointcat an integer specifying the character size for the points showing the categories, used with par("cex")*cpointcat

labelrowprof a vector of strings of characters for the labels of profiles (rows of dfrowprof)

clabelrowprof an integer specifying the character size for the labels of profiles used with par("cex")*clabelrowprof

cpointrowprof an integer specifying the character size for the points representative of the profiles used with par("cex")*cpointrowprof

pchrowprof either an integer specifying a symbol or a single character to be used for the profile labels

coulrowprof a vector of colors used for ellipses, possibly recycled

proba a value lying between 0.500 and 0.999 to draw a confidence interval

n.sample a vector containing the sample size, possibly recycled. Used n.sample = 0 if the profiles are not issued from a multinomial distribution and that confidence intervals have no sense.

axesell a logical value indicating whether the ellipse axes should be drawn

... further arguments passed from the s.label for the initial scatter plot.
**Value**

Returns in a hidden way a list of three components:

- `tra` a vector with two values giving the done original translation.
- `ell` a matrix, with 5 columns and for rows the number of profiles, giving the means, the variances and the covariance of the profile for the used numerical codes (column of `dfxy`)
- `call` the matched call

**Author(s)**

Daniel Chessel

**Examples**

```r
par(mfrow = c(2,2))
par(mar = c(0.1,0.1,0.1,0.1))
proba <- matrix(c(0.49,0.47,0.04,0.4,0.3,0.3,0.05,0.05,0.9,0.05,0.7,0.25), ncol = 3, byrow = TRUE)
proba.df <- as.data.frame(proba)
names(proba.df) <- c("A","B","C") ; row.names(proba.df) <- c("P1","P2","P3","P4")
w.proba <- triangle.plot(proba.df, clab = 2, show = FALSE)
box()

w.tri = data.frame(x = c(-sqrt(1/2),sqrt(1/2),0), y = c(-1/sqrt(6),-1/sqrt(6),2/sqrt(6)))
L3 <- c("A","B","C")
row.names(w.tri) <- L3
s.multinom(w.tri, proba.df, n.sample = 0, coulrowprof = "black", clabelrowprof = 1.5)
s.multinom(w.tri, proba.df, n.sample = 30, coul = palette()[5])
s.multinom(w.tri, proba.df, n.sample = 60, coul = palette()[6], add.p = TRUE)
s.multinom(w.tri, proba.df, n.sample = 120, coul = grey(0.8), add.p = TRUE)

print(s.multinom(w.tri, proba.df[-3,], n.sample = 0, translate = TRUE)$tra)
```

**Description**

performs the scatter diagram with trajectories.

**Usage**

```r
s.traject(dfxy, fac = factor(rep(1, nrow(dfxy))),
           ord = (1:length(fac)), xax = 1, yax = 2, label = levels(fac),
           clabel = 1, cpoint = 1, pch = 20, xlim = NULL, ylim = NULL,
           grid = TRUE, addaxes = TRUE, edge = TRUE, origin = c(0,0),
           include.origin = TRUE, sub = "", csub = 1, possub = "bottomleft",
           cgrid = 1, pixmap = NULL, contour = NULL, area = NULL, add.plot = FALSE)
```
Arguments

- `dfxy`: a data frame containing two columns for the axes
- `fac`: a factor partitioning the rows of the data frame in classes
- `ord`: a vector of length equal to `fac`. The trajectory is drawn in an ascending order of the `ord` values
- `xax`: the column number for the x-axis
- `yax`: the column number for the y-axis
- `label`: a vector of strings of characters for the point labels
- `clabel`: if not NULL, a character size for the labels, used with `par("cex")*clabel`
- `cpoint`: a character size for plotting the points, used with `par("cex")*cpoint`. If zero, no points are drawn
- `pch`: if `cpoint > 0`, an integer specifying the symbol or the single character to be used in plotting points
- `xlim`: the ranges to be encompassed by the x, if `NULL` they are computed
- `ylim`: the ranges to be encompassed by the y, if `NULL` they are computed
- `grid`: a logical value indicating whether a grid in the background of the plot should be drawn
- `addaxes`: a logical value indicating whether the axes should be plotted
- `edge`: if `TRUE` the arrows are plotted, otherwise only the segments
- `origin`: the fixed point in the graph space, for example c(0,0) the origin axes
- `include.origin`: a logical value indicating whether the point "origin" should be belonged to the graph space
- `sub`: a string of characters to be inserted as legend
- `csub`: a character size for the legend, used with `par("cex")*csub`
- `possub`: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- `cgrid`: a character size, parameter used with `par("cex")*cgrid` to indicate the mesh of the grid
- `pixmap`: an object 'pixmap' displayed in the map background
- `contour`: a data frame with 4 columns to plot the contour of the map: each row gives a segment (x1,y1,x2,y2)
- `area`: a data frame of class 'area' to plot a set of surface units in contour
- `add.plot`: if `TRUE` uses the current graphics window

Value

The matched call.

Author(s)

Daniel Chessel
Examples

```r
if(!adegraphicsLoaded()) {
  rw <- function(a) {
    x <- 0
    for(i in 1:49) x <- c(x, x[length(x)] + runif(1, -1, 1))
    x
  }
  y <- unlist(lapply(1:5, rw))
  x <- unlist(lapply(1:5, rw))
  z <- gl(5, 50)
  s.traject(data.frame(x, y), z, edge = FALSE)
}
```

---

**s.value**

*Representation of a value in a graph*

Description

performs the scatter diagram with the representation of a value for a variable

Usage

```r
s.value(dfxy, z, xax = 1, yax = 2, method = c("squaresize", "greylevel"),
    zmax=NULL, csize = 1, cpoint = 0, pch = 20, clegend = 0.75, neig = NULL,
    cneig = 1, xlim = NULL, ylim = NULL, grid = TRUE, addaxes = TRUE,
    cgrid = 0.75, include.origin = TRUE, origin = c(0,0), sub = "",
    csub = 1, possub = "topleft", pixmap = NULL, contour = NULL,
    area = NULL, add.plot = FALSE)
```

Arguments

- `dfxy`: a data frame with two coordinates
- `z`: a vector of the values corresponding to the rows of `dfxy`
- `xax`: column for the x axis
- `yax`: column for the y axis
- `method`: a string of characters
  "squaresize" gives black squares for positive values and white for negative values with a proportional area equal to the absolute value.
  "greylevel" gives squares of equal size with a grey level proportional to the value. By default the first choice
- `zmax`: a numeric value, equal by default to `max(abs(z))`, can be used to impose a common scale of the size of the squares to several drawings in the same device.
- `csize`: a size coefficient for symbols
- `cpoint`: a character size for plotting the points, used with `par("cex")*cpoint`. If zero, no points are drawn
pch if cpoint > 0, an integer specifying the symbol or the single character to be used in plotting points

clegend a character size for the legend used by par("cex")*clegend

neig a neighbouring graph

cneig a size for the neighbouring graph lines used with par("lwd")*cneig

xlim the ranges to be encompassed by the x, if NULL they are computed

ylim the ranges to be encompassed by the y, if NULL they are computed

grid a logical value indicating whether a grid in the background of the plot should be drawn

addaxes a logical value indicating whether the axes should be plotted

cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid

include.origin a logical value indicating whether the point "origin" should be belonged to the graph space

origin the fixed point in the graph space, for example c(0,0) the origin axes

sub a string of characters to be inserted as legend

csub a character size for the legend, used with par("cex")*csub

possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

pixmap an object 'pixmap' displayed in the map background

contour a data frame with 4 columns to plot the contour of the map : each row gives a segment (x1,y1,x2,y2)

area a data frame of class 'area' to plot a set of surface units in contour

add.plot if TRUE uses the current graphics window

Value

The matched call.

Author(s)

Daniel Chessel

Examples

if(!adegraphicsLoaded()) {
  xy <- cbind.data.frame(x = runif(500), y = runif(500))
  z <- rnorm(500)
  s.value(xy, z)

  s.value(xy, z, method = "greylevel")
}

data(rpjdl)
  fau.coa <- dudi.coa(rpjdl$fau, scan = FALSE, nf = 3)
  s.value(fau.coa$li, fau.coa$li[,3], csi = 0.75, cleg = 0.75)
data(irishdata)
par(mfrow = c(3, 4))
irq0 <- data.frame(scale(irishdata$tab, scale = TRUE))
for (i in 1:12) {
  z <- irq0[, i]
  nam <- names(irq0)[i]
  s.value(irishdata$xy, z, area = irishdata$area, csi = 3,
          csub = 2, sub = nam, cleg = 1.5, cgrid = 0, inc = FALSE,
          xlim = c(16, 205), ylim = c(-50, 268), adda = FALSE, grid = FALSE)
}

santacatalina

**Indirect Ordination**

**Description**

This data set gives the densities per hectare of 11 species of trees for 10 transects of topographic moisture values (mean of several stations per class).

**Usage**

data(santacatalina)

**Format**

a data frame with 11 rows and 10 columns

**Source**


**Examples**

data(santacatalina)
coa1 <- dudi.coa(log(santacatalina + 1), scan = FALSE) # 2 factors

if(adegraphicsLoaded()) {
  g1 <- table.value(log(santacatalina + 1), plot = FALSE)
  g2 <- table.value(log(santacatalina + 1)[, sample(10)], plot = FALSE)
  g3 <- table.value(log(santacatalina + 1)[order(coa1$li[, 1]), order(coa1$co[, 1])], plot = FALSE)
  g4 <- scatter(coa1, posi = "bottomright", plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
table.value(log(santacatalina + 1))
table.value(log(santacatalina + 1)[, sample(10)])
table.value(log(santacatalina + 1)[order(coa1$li[, 1]), order(coa1$co[, 1])])
scatter(coa, posi = "bottomright")
par(mfrow = c(1, 1))
}

sarcelles

---

**Array of Recapture of Rings**

**Description**

The data frame `sarcelles$tab` contains the number of the winter teals (`Anas C. Crecca`) for which the ring was retrieved in the area `i` during the month `j` (`n=3049`).

**Usage**

```r
data(sarcelles)
```

**Format**

sarcelles is a list with the following components:

- `tab` a data frame with 14 rows-areas and 12 columns-months
- `xy` a data frame with the 2 spatial coordinates of the 14 region centers
- `neig` the neighbouring graph between areas, object of the class `neig`
- `col.names` a vector containing the month items
- `nb` a neighborhood object (class `nb` defined in package `spdep`)

**Source**


**Examples**

```r
## Not run:
if(!adeGraphicsLoaded()) {
  # depends of pixmap
  if(requireNamespace("pixmap", quietly = TRUE)) {
    bkpng.pnm <- pixmap:::read.pnm(system.file("pictures/sarcelles.pnm", package = "ade4"))
    data(sarcelles)
    par(mfrow = c(4, 3))
    for(i in 1:12) {
      s.distri(sarcelles$xy, sarcelles$tab[, i], pixmap = bkpng.pnm, sub = sarcelles$col.names[i], clab = 0, csub = 2)
      s.value(sarcelles$xy, sarcelles$tab[, i], add.plot = TRUE, cleg = 0)
    }
    par(mfrow = c(1, 1))
  }
}
## End(Not run)
```
**Description**

These utility functions compute (weighted) means, variances and covariances for dataframe partitioned by a factor. The scale transforms a numeric matrix in a centred and scaled matrix for any weighting.

**Usage**

```r
covwt(x, wt, na.rm = FALSE)
varwt(x, wt, na.rm = FALSE)
scalewt(df, wt = rep(1/nrow(df), nrow(df)), center = TRUE, scale = TRUE)
meanfacwt(df, fac = NULL, wt = rep(1/nrow(df), nrow(df)), drop = FALSE)
varfacwt(df, fac = NULL, wt = rep(1/nrow(df), nrow(df)), drop = FALSE)
covfacwt(df, fac = NULL, wt = rep(1/nrow(df), nrow(df)), drop = FALSE)
scalefacwt(df, fac = NULL, wt = rep(1/nrow(df), nrow(df)), scale = TRUE, drop = FALSE)
```

**Arguments**

- `x`: a numeric vector (`varwt`) or a matrix (`covwt`) containing the data.
- `na.rm`: a logical value indicating whether NA values should be stripped before the computation proceeds.
- `df`: a matrix or a dataframe containing the data.
- `fac`: a factor partitioning the data.
- `wt`: a numeric vector of weights.
- `drop`: a logical value indicating whether unused levels should be kept.
- `scale`: a logical value indicating whether data should be scaled or not.
- `center`: a logical value indicating whether data should be centered or not.

**Details**

Functions return biased estimates of variances and covariances (i.e. divided by n and not n-1)

**Value**

For `varwt`, the weighted variance. For `covwt`, the matrix of weighted co-variances. For `scalewt`, the scaled dataframe. For other function a list (if `fac` is not null) of dataframes with appropriate values.

**Author(s)**

Stéphane Dray <stephane.dray@univ-lyon1.fr>
Examples

data(meau)
w <- rowSums(meau$spe)
varwt(meau$env, w)
varfacwt(meau$env, wt = w)
varfacwt(meau$env, wt = w, fac = meau$design$season)
covfacwt(meau$env, wt = w, fac = meau$design$season)
scalewt(meau$env, wt = w)

scatter  
Graphical representation of the outputs of a multivariate analysis

Description

scatter is a generic function that has methods for the classes coa, dudi, fca, acm and pco. It plots the outputs of a multivariate analysis by representing simultaneously the rows and the columns of the original table (biplot). The function biplot returns exactly the same representation. The function screeplot represents the amount of inertia (usually variance) associated to each dimension.

Usage

scatter(x, ...)
## S3 method for class 'dudi'
biplot(x, ...)
## S3 method for class 'dudi'
screep(x, npcs = length(x$eig), type = c("barplot", "lines"),
       main = deparse(substitute(x)), col = c(rep("black", x$nf),
       rep("grey", npcs - x$nf)), ...)

Arguments

x an object of the class dudi containing the outputs of a multivariate analysis
npcs the number of components to be plotted
type the type of plot
main the title of the plot
col a vector of colors
... further arguments passed to or from other methods

Author(s)

Daniel Chessel
Stéphane Dray <stephane.dray@univ-lyon1.fr>
See Also

`s.arrow, s.chull, s.class, s.corcircle, s.distri, s.label, s.match, s.traject, s.value, add.scatter`

Examples

data(rpjd1)
rpjd1.coa <- dudi.coa(rpjd1$fau, scannf = FALSE, nf = 4)
screepplot(rpjd1.coa)
biplot(rpjd1.coa)

---

**scatter.acm**

*Plot of the factorial maps in a Multiple Correspondence Analysis*

Description

performs the scatter diagrams of a Multiple Correspondence Analysis.

Usage

```r
## S3 method for class 'acm'
scatter(x, xax = 1, yax = 2, mfrow = NULL, csub = 2, possub = "topleft", ...)
```

Arguments

- `x`  an object of class acm
- `xax`  the column number for the x-axis
- `yax`  the column number for the y-axis
- `mfrow`  a vector of the form "c(nr,nc)", if NULL (the default) is computed by n2mfrow
- `csub`  a character size for the legend, used with par("cex")*csub
- `possub`  a string of characters indicating the legend position ("topleft", "topright", "bottomleft", "bottomright") in a array of figures
- `...`  further arguments passed to or from other methods

Author(s)

Daniel Chessel

Examples

data(lascaux)
if(adegraphicsLoaded()) {
  plot(dudi.acm(lascaux$ornem, sca = FALSE))
} else {
  scatter(dudi.acm(lascaux$ornem, sca = FALSE), csub = 3)
}
scatter.coa

Plot of the factorial maps for a correspondence analysis

Description

performs the scatter diagrams of a correspondence analysis.

Usage

## S3 method for class 'coa'
scatter(x, xax = 1, yax = 2, method = 1:3, clab.row = 0.75,
       clab.col = 1.25, posieig = "top", sub = NULL, csub = 2, ...)

Arguments

x an object of class coa
xax the column number for the x-axis
yax the column number for the y-axis
method an integer between 1 and 3
   1 Rows and columns with the coordinates of lambda variance
   2 Columns variance 1 and rows by averaging
   3 Rows variance 1 and columns by averaging
clab.row a character size for the rows
clab.col a character size for the columns
posieig if "top" the eigenvalues bar plot is upside, if "bottom" it is downside, if "none" no plot
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
... further arguments passed to or from other methods

Author(s)

Daniel Chessel

References

Examples

```r
data(housetasks)
w <- dudi.coa(housetasks, scan = FALSE)
if(adegraphicsLoaded()) {
  g1 <- scatter(w, method = 1, psub.text = "1 / Standard", posieig = "none", plot = FALSE)
g2 <- scatter(w, method = 2, psub.text = "2 / Columns -> averaging -> Rows",
                   posieig = "none", plot = FALSE)
g3 <- scatter(w, method = 3, psub.text = "3 / Rows -> averaging -> Columns ",
                   posieig = "none", plot = FALSE)
G <- ADEgS(list(g1, g2, g3), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  scatter(w, method = 1, sub = "1 / Standard", posieig = "none")
  scatter(w, method = 2, sub = "2 / Columns -> averaging -> Rows", posieig = "none")
  scatter(w, method = 3, sub = "3 / Rows -> averaging -> Columns ", posieig = "none")
  par(mfrow = c(1, 1))
}
```

---

**scatter.dudi**

![Plot of the Factorial Maps](image)

**Description**

performs the scatter diagrams of objects of class dudi.

**Usage**

```r
## S3 method for class 'dudi'
scatter(x, xax = 1, yax = 2, clab.row = 0.75, clab.col = 1,
        permute = FALSE, posieig = "top", sub = NULL, ...)
```

**Arguments**

- `x`: an object of class dudi
- `xax`: the column number for the x-axis
- `yax`: the column number for the y-axis
- `clab.row`: a character size for the rows
- `clab.col`: a character size for the columns
- `permute`: if FALSE, the rows are plotted by points and the columns by arrows. If TRUE it is the opposite.
- `posieig`: if "top" the eigenvalues bar plot is upside, if "bottom" it is downside, if "none" no plot
- `sub`: a string of characters to be inserted as legend
- `...`: further arguments passed to or from other methods
scatter.dudi is a factorial map of individuals and the projection of the vectors of the canonical basis multiplied by a constant of rescaling. In the eigenvalues bar plot, the used axes for the plot are in black, the other kept axes in grey and the other in white.

The permute argument can be used to choose between the distance biplot (default) and the correlation biplot (permute = TRUE).

Author(s)
Daniel Chessel

Examples

```r
data(deug)
sscatter(dd1 <- dudi.pca(deug$tab, scannf = FALSE, nf = 4),
        posieig = "bottomright")
```

data(rhone)
dd1 <- dudi.pca(rhone$tab, nf = 4, scann = FALSE)
if(adegraphicsLoaded()) {
  scatter(dd1, row.psub.text = "Principal component analysis")
} else {
  scatter(dd1, sub = "Principal component analysis")
}
```

Description
performs the scatter diagrams of a fuzzy correspondence analysis.

Usage

```r
## S3 method for class 'fca'
sscatter(x, xax = 1, yax = 2, clab.moda = 1, labels = names(x$tab),
         sub = NULL, csub = 2, ...)
```

Arguments

- `x` an object of class `fca`
- `xax` the column number for the x-axis
- `yax` the column number for the y-axis
- `clab.moda` the character size to write the modalities
- `labels` a vector of strings of characters for the labels of the modalities
- `sub` a vector of strings of characters to be inserted as legend in each figure
- `csub` a character size for the legend, used with par("cex")*csub
- `...` further arguments passed to or from other methods
**Examples**

```r
data(coleo)
coleo.fuzzy <- prep.fuzzy.var(coleo$tab, coleo$col.blocks)
fca1 <- dudi.fca(coleo.fuzzy, sca = FALSE, nf = 3)

if(adegraphicsLoaded()) {
  plot(fca1)
} else {
  scatter(fca1, labels = coleo$moda.names, clab.moda = 1.5,
         sub = names(coleo$col.blocks), csub = 3)
}
```

**Description**

These are utilities used in graphical functions.

**Details**

The functions `scatter` use some utilities functions:

- `scatterutil.base` defines the layer of the plot for all scatters
- `scatterutil.sco` defines the layer of the plot for sco functions
- `scatterutil.chull` plots the polygons of the external contour
- `scatterutil.eigen` plots the eigenvalues bar plot
- `scatterutil.ellipse` plots an inertia ellipse for a weighting distribution
- `scatterutil.eti.circ` puts labels on a correlation circle
- `scatterutil.eti` puts labels centred on the points
- `scatterutil.grid` plots a grid and adds a legend
- `scatterutil.legend.bw.square` puts a legend of values by square size
- `scatterutil.legend.square.grey` puts a legend by squares and grey levels
- `scatterutil.legendgris` adds a legend of grey levels for the areas
- `scatterutil.scaling` to fit a plot on a background bipmap
- `scatterutil.star` plots a star for a weighting distribution
- `scatterutil.sub` adds a string of characters in sub-title of a graph
- `scatterutil.convrot90` is used to rotate labels
Author(s)
Daniel Chessel, Stéphane Dray <stephane.dray@univ-lyon1.fr>

See Also
s.arrow, s.chull, s.class, s.corcircle, s.distri, s.label, s.match, s.traject, s.value, add.scatter

Examples
par(mfrow = c(3,3))
plot.new()
ade4::scatterutil.legendgris(1:20, 4, 1.6)

plot.new()
ade4::scatterutil.sub("lkn5555555555lkn", csub = 2, possub = "bottomleft")
ade4::scatterutil.sub("lkn5555555555lkn", csub = 1, possub = "topleft")
ade4::scatterutil.sub("jdjjl", csub = 3, possub = "topright")
ade4::scatterutil.sub("***", csub = 2, possub = "bottomright")

x <- c(0.5, 0.2, -0.5, -0.2); y <- c(0.2, 0.5, -0.2, -0.5)
eti <- c("toto", "kjbk", "gdgiglg1", "sdg")
plot(x, y, xlim = c(-1,1), ylim = c(-1,1))
ade4::scatterutil.eti.circ(x, y, eti, 2.5)
abline(0, 1, lty = 2); abline(0, -1, lty = 2)

x <- c(0.5, 0.2, -0.5, -0.2); y <- c(0.2, 0.5, -0.2, -0.5)
eti <- c("toto", "kjbk", "gdgiglg1", "sdg")
plot(x, y, xlim = c(-1,1), ylim = c(-1,1))
ade4::scatterutil.eti(x, y, eti, 1.5)

plot(runif(10,-3,5), runif(10,-1,1), asp = 1)
ade4::scatterutil.grid(2)
abline(h = 0, v = 0, lwd = 3)

x <- runif(10,0,1); y <- rnorm(10); z <- rep(1,10)
plot(x,y); ade4::scatterutil.star(x, y, z, 0.5)
plot(x,y); ade4::scatterutil.star(x, y, z, 1)

x <- c(runif(10,0,0.5), runif(10,0.5,1))
y <- runif(20)
plot(x, y, asp = 1) # asp=1 is essential to have perpendicular axes
ade4::scatterutil.ellipse(x, y, rep(c(1,0), c(10,10)), cell = 1.5, ax = TRUE)
ade4::scatterutil.ellipse(x, y, rep(c(0,1), c(10,10)), cell = 1.5, ax = TRUE)

x <- c(runif(100,0,0.75), runif(100,0.25,1))
y <- c(runif(100,0,0.75), runif(100,0.25,1))
z <- factor(rep(c(1,2), c(100,100)))
plot(x, y, pch = rep(c(1,20), c(100,100)))
ade4::scatterutil.chull(x, y, z, opt = c(0.25,0.5,0.75,1))
par(mfrow = c(1,1))
**sco.boxplot**

*Representation of the link between a variable and a set of qualitative variables*

**Description**

represents the link between a variable and a set of qualitative variables.

**Usage**

```r
sco.boxplot(score, df, labels = names(df), clabel = 1, xlim = NULL,
grid = TRUE, cgrid = 0.75, include.origin = TRUE, origin = 0,
sub = NULL, csub = 1)
```

**Arguments**

- `score`: a numeric vector
- `df`: a data frame with only factors
- `labels`: a vector of strings of characters for the labels of variables
- `clabel`: if not NULL, a character size for the labels, used with `par("cex")`*clabel
- `xlim`: the ranges to be encompassed by the x axis, if NULL they are computed
- `grid`: a logical value indicating whether the scale vertical lines should be drawn
- `cgrid`: a character size, parameter used with `par("cex")`*cgrid to indicate the mesh of the scale
- `include.origin`: a logical value indicating whether the point "origin" should be belonged to the graph space
- `origin`: the fixed point in the graph space, for example 0 the origin axis
- `sub`: a string of characters to be inserted as legend
- `csub`: a character size for the legend, used with `par("cex")`*csub

**Author(s)**

Daniel Chessel

**Examples**

```r
w1 <- rnorm(100,-1)
w2 <- rnorm(100)
w3 <- rnorm(100,1)
f1 <- gl(3,100)
f2 <- gl(30,10)
sco.boxplot(c(w1,w2,w3), data.frame(f1,f2))

data(banque)
banque.acm <- dudi.acm(banque, scan = FALSE, nf = 4)
```
par(mfrow = c(1,3))
sco.boxplot(banque.acm$l1[,1], banque[,1:7], clab = 1.8)
sco.boxplot(banque.acm$l1[,1], banque[,8:14], clab = 1.8)
sco.boxplot(banque.acm$l1[,1], banque[,15:21], clab = 1.8)
par(mfrow = c(1,1))

sco.class

1D plot of a numeric score and a factor with labels

Description

Draws evenly spaced labels, each label linked to the corresponding values of the levels of a factor.

Usage

sco.class(score, fac, label = levels(fac), clabel = 1, horizontal = TRUE, reverse = FALSE, pos.lab = 0.5, pch = 20, cpoint = 1, boxes = TRUE, col = rep(1, length(levels(fac))), lim = NULL, grid = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0, 0), sub = "", csub = 1.25, possub = "bottomleft")

Arguments

score      a numeric vector
fac         a factor
label      labels for the levels of the factor
clabel      a character size for the labels, used with par("cex")*clabel
horizontal  logical. If TRUE, the plot is horizontal
reverse     logical. If horizontal = TRUE and reverse=TRUE, the plot is at the bottom, if reverse = FALSE, the plot is at the top. If horizontal = FALSE, the plot is at the right (TRUE) or at the left (FALSE).
pos.lab     a values between 0 and 1 to manage the position of the labels.
pch         an integer specifying the symbol or the single character to be used in plotting points
cpoint      a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
boxes       if TRUE, labels are framed
col         a vector of colors used to draw each class in a different color
lim          the range for the x axis or y axis (if horizontal = FALSE), if NULL, they are computed
grid        a logical value indicating whether a grid in the background of the plot should be drawn
cgrid       a character size, parameter used with par("cex")* cgrid to indicate the mesh of the grid
include.origin  a logical value indicating whether the point "origin" should belong to the plot
origin        the fixed point in the graph space, for example c(0,0) the origin axes
sub           a string of characters to be inserted as legend
csub          a character size for the legend, used with par("cex")*csub
possub        a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

Value
The matched call.

Author(s)
Stéphane Dray <stephane.dray@univ-lyon1.fr>

Examples

```r
data(meau)
enpca <- dudi.pca(meau$env, scannf=FALSE)
par(mfrow=c(2,1))
sco.class(envpca$li[,1],meau$design$season, col = 1:6)
sco.class(envpca$li[,1],meau$design$season, col = 1:4, reverse = TRUE)
```

---

**sco.distri**  
*Representation by mean- standard deviation of a set of weight distributions on a numeric score*

**Description**
represents the mean- standard deviation of a set of weight distributions on a numeric score.

**Usage**
```
sco.distri(score, df, y.rank = TRUE, csize = 1, labels = names(df),
clabel = 1, xlim = NULL, grid = TRUE, cgrid = 0.75,
include.origin = TRUE, origin = 0, sub = NULL, csub = 1)
```

**Arguments**
- **score**: a numeric vector
- **df**: a data frame with only positive or null values
- **y.rank**: a logical value indicating whether the means should be classified in ascending order
- **csize**: an integer indicating the size segment
- **labels**: a vector of strings of characters for the labels of the variables
- **clabel**: if not NULL, a character size for the labels, used with par("cex")*clabel
sco.distri

xlim the ranges to be encompassed by the x axis, if NULL they are computed
grid a logical value indicating whether the scale vertical lines should be drawn
cgrid a character size, parameter used with par("cex")*cgrid to indicate the mesh of the scale
include.origin a logical value indicating whether the point "origin" should be belonged to the graph space
origin the fixed point in the graph space, for example c(0,0) the origin axes
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub

Value
returns an invisible data.frame with means and variances

Author(s)
Daniel Chessel

Examples

```r
if(!adegraphicsLoaded()) {
  w <- seq(-1, 1, le = 200)
distri <- data.frame(lapply(1:50,
    function(x) sample((200:1)) * ((w >= (- x / 50)) & (w <= x / 50)))
names(distri) <- paste("w", 1:50, sep = "")
par(mfrow = c(1, 2))
sco.distri(w, distri, csi = 1.5)
sco.distri(w, distri, y.rank = FALSE, csi = 1.5)
par(mfrow = c(1, 1))
}
data(rpjdl)
coa2 <- dudi.coa(rpjdl$fau, FALSE)
sco.distri(coa2$li[, 1], rpjdl$fau, lab = rpjdl$frlab, clab = 0.8)

data(doubs)
par(mfrow = c(2, 2))
poi.coa <- dudi.coa(doubs$fish, scann = FALSE)
sco.distri(poi.coa$li[, 1], doubs$fish)
poi.nsc <- dudi.nsc(doubs$fish, scann = FALSE)
sco.distri(poi.nsc$li[, 1], doubs$fish)
s.label(poi.coa$li)
s.label(poi.nsc$li)

data(rpjdl)
fau.coa <- dudi.coa(rpjdl$fau, scann = FALSE)
sco.distri(fau.coa$li[, 1], rpjdl$fau)
fau.nsc <- dudi.nsc(rpjdl$fau, scann = FALSE)
sco.distri(fau.nsc$li[, 1], rpjdl$fau)
s.label(fau.coa$li)
s.label(fau.nsc$li)
```

par(mfrow = c(1, 1))

Relationship between one score and qualitative variables

Description

Draws Gauss curves with the same mean and variance as the scores of individuals belonging to categories of several qualitative variables.

Usage

sco.gauss(score, df, xlim = NULL, steps = 200, ymax = NULL, sub = names(df), csub = 1.25, possub = "topleft", legen = TRUE, label = row.names(df), clabel = 1, grid = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0, 0))

Arguments

score  a numeric vector
df  a dataframe containing only factors, number of rows equal to the length of the score vector
xlim  starting point and end point for drawing the Gauss curves
steps  number of segments for drawing the Gauss curves
ymax  max ordinate for all Gauss curves. If NULL, ymax is computed and different for each factor
sub  vector of strings of characters for the labels of qualitative variables
csub  character size for the legend
possub  a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
legen  if TRUE, the first graphic of the series displays the score with evenly spaced labels (see sco.label)
label  labels for the score
clabel  a character size for the labels, used with par("cex")*clabel
grid  a logical value indicating whether a grid in the background of the plot should be drawn
cgrid  a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
include.origin  a logical value indicating whether the point "origin" should belong to the plot
origin  the fixed point in the graph space, for example c(0,0) the origin axes
**Details**

Takes one vector containing quantitative values (score) and one dataframe containing only factors that give categories to which the quantitative values belong. Computes the mean and variance of the values in each category of each factor, and draws a Gauss curve with the same mean and variance for each category of each factor. Can optionally set the start and end point of the curves and the number of segments. The max ordinate (ymax) can also be set arbitrarily to set a common max for all factors (else the max is different for each factor).

**Value**

The matched call.

**Author(s)**

Jean Thioulouse, Stéphane Dray <stephane.dray@univ-lyon1.fr>

**Examples**

```r
data(meau)
enpca <- dudi.pca(meau$env, scannf=FALSE)
dffac <- cbind.data.frame(meau$design$season, meau$design$site)
sco.gauss(envpca$li[,1], dffac, clabel = 2, csub = 2)
```

---

**sco.label**

1D plot of a numeric score with labels

**Description**

Draws evenly spaced labels, each label linked to the corresponding value of a numeric score.

**Usage**

```r
sco.label(score, label = names(score), clabel = 1, horizontal = TRUE, reverse = FALSE, pos.lab = 0.5, pch = 20, cpoint = 1, boxes = TRUE, lim = NULL, grid = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0, 0), sub = "", csub = 1.25, possub = "bottomleft")
```

**Arguments**

- **score**: a numeric vector
- **label**: labels for the score
- **clabel**: a character size for the labels, used with par("cex")*clabel
- **horizontal**: logical. If TRUE, the plot is horizontal
- **reverse**: logical. If horizontal = TRUE and reverse=TRUE, the plot is at the bottom, if reverse = FALSE, the plot is at the top. If horizontal = FALSE, the plot is at the right (TRUE) or at the left (FALSE).
pos.lab  a values between 0 and 1 to manage the position of the labels.
pch  an integer specifying the symbol or the single character to be used in plotting points
cpoint  a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
boxes  if TRUE, labels are framed
lim  the range for the x axis or y axis (if horizontal = FALSE), if NULL, they are computed
grid  a logical value indicating whether a grid in the background of the plot should be drawn
cgrid  a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid
include.origin  a logical value indicating whether the point "origin" should belong to the plot
origin  the fixed point in the graph space, for example c(0,0) the origin axes
sub  a string of characters to be inserted as legend
csub  a character size for the legend, used with par("cex")*csub
possub  a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

Value
The matched call.

Author(s)
Stéphane Dray <stephane.dray@univ-lyon1.fr>, Jean Thioulouse

Examples

```r
data(meau)
envpca <- dudi.pca(meau$env, scannf=FALSE)
par(mfrow=c(2,1))
sco.label(envpca$l1[,1], row.names(envpca$l1), lim=c(-1,3.5))
sco.label(envpca$co[,1], row.names(envpca$co), reverse = TRUE, lim=c(-1,3.5))
```

---

sco.match  1D plot of a pair of numeric scores with labels

Description

Draws evenly spaced labels, each label linked to the corresponding values of two numeric score.
sco.match

Usage

sco.match(score1, score2, label = names(score1), clabel = 1,
horizontal = TRUE, reverse = FALSE, pos.lab = 0.5, wmatch = 3,
pch = 20, cpoint = 1, boxes = TRUE, lim = NULL, grid = TRUE,
cgrid = 1, include.origin = TRUE, origin = c(0, 0), sub = "",
csub = 1.25, possub = "bottomleft")

Arguments

score1 a numeric vector
score2 a numeric vector
label labels for the score
clabel a character size for the labels, used with par("cex")*cex
horizontal logical. If TRUE, the plot is horizontal
reverse logical. If horizontal = TRUE and reverse=TRUE, the plot is at the bottom, if reverse = FALSE, the plot is at the top. If horizontal = FALSE, the plot is at the right (TRUE) or at the left (FALSE).
pos.lab a values between 0 and 1 to manage the position of the labels.
wmatch a numeric values to specify the width of the matching region in the plot. The width is equal to wmatch * the height of character
pch an integer specifying the symbol or the single character to be used in plotting points
cpoint a character size for plotting the points, used with par("cex")*cex. If zero, no points are drawn
boxes if TRUE, labels are framed
lim the range for the x axis or y axis (if horizontal = FALSE), if NULL, they are computed
grid a logical value indicating whether a grid in the background of the plot should be drawn
cgrid a character size, parameter used with par("cex")*cexgrid to indicate the mesh of the grid
include.origin a logical value indicating whether the point "origin" should belong to the plot
origin the fixed point in the graph space, for example c(0,0) the origin axes
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*cex
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

Value

The matched call.
sco.quant

Graph to Analyse the Relation between a Score and Quantitative Variables

Description
represents the graphs to analyse the relation between a score and quantitative variables.

Usage
sco.quant (score, df, fac = NULL, clabel = 1, abline = FALSE, sub = names(df), csub = 2, possub = "topleft")

Arguments
score            a numeric vector
df               a data frame which rows equal to the score length
fac              a factor with the same length than the score
clabel           character size for the class labels (if any) used with par("cex")*clabel
abline           a logical value indicating whether a regression line should be added
sub              a vector of strings of characters for the labels of variables
csub             a character size for the legend, used with par("cex")*csub
possub           a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

Author(s)
Daniel Chessel

Examples
w <- runif(100, -5, 10)
wX <- data.frame(matrix(w + rnorm(900, sd = (1:900) / 100), 100, 9))
sco.quant(w, wX, fac = fw, abline = TRUE, clab = 2, csub = 3)
**Description**

score is a generic function. It proposes methods for the objects 'coa', 'acm', 'mix', 'pca'.

**Usage**

```r
score(x, ...) 
scoreutil.base(y, xlim, grid, cgrid, include.origin, origin, sub, csub)
```

**Arguments**

- `x`: an object used to select a method
- `...`: further arguments passed to or from other methods
- `y`: a numeric vector
- `xlim`: the ranges to be encompassed by the x axis, if NULL they are computed
- `grid`: a logical value indicating whether the scale vertical lines should be drawn
- `cgrid`: a character size, parameter used with `par("cex")*cgrid` to indicate the mesh of the scale
- `include.origin`: a logical value indicating whether the point "origin" should be belonged to the graph space
- `origin`: the fixed point in the graph space, for example 0 the origin axis
- `sub`: a string of characters to be inserted as legend
- `csub`: a character size for the legend, used with `par("cex")*csub`

**Details**

scoreutil.base is a utility function - not for the user - to define the bottom of the layout of all score.

**Author(s)**

Daniel Chessel

**See Also**

- `sco.boxplot`
- `sco.distri`
- `sco.quant`
Examples

```r
## Not run:
par(mar = c(1, 1, 1, 1))
ade4:::scoreutil.base (runif(20, 3, 7), xlim = NULL, grid = TRUE, cgrid = 0.8,
include.origin = TRUE, origin = 0, sub = "Uniform", csub = 1)
## End(Not run)
# returns the value of the user coordinate of the low line.
# The user window id defined with c(0,1) in ordinate.
# box()
```

---

**score.acm**

*Graphs to study one factor in a Multiple Correspondence Analysis*

## Description

performs the canonical graph of a Multiple Correspondence Analysis.

## Usage

```r
## S3 method for class 'acm'
score(x, xax = 1, which.var = NULL, mfrow = NULL,
sub = names(oritab), csub = 2, possub = "topleft", ...)
```

## Arguments

- `x`: an object of class `acm`
- `xax`: the column number for the used axis
- `which.var`: the numbers of the kept columns for the analysis, otherwise all columns
- `mfrow`: a vector of the form "c(nr,nc)", otherwise computed by a special own function `n2mfrow`
- `sub`: a vector of strings of characters to be inserted as sub-titles, otherwise the variable names of the initial array
- `csub`: a character size for the sub-titles
- `possub`: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
- `...`: further arguments passed to or from other methods

## Author(s)

Daniel Chessel

## Examples

```r
data(banque)
banque.acm <- dudi.acm(banque, scann = FALSE, nf = 3)
score(banque.acm, which = which(banque.acm$cr[, 1] > 0.2))
```
Reciprocal scaling after a correspondence analysis

Description

performs the canonical graph of a correspondence analysis.

Usage

## S3 method for class 'coa'
score(x, xax = 1, dotchart = FALSE, clab.r = 1, clab.c = 1, 
csub = 1, cpoi = 1.5, cet = 1.5, ...)
reciprocal.coa(x)

Arguments

x an object of class coa
xax the column number for the used axis
dotchart if TRUE the graph gives a "dual scaling", if FALSE a "reciprocal scaling"
clab.r a character size for row labels
clab.c a character size for column labels
csub a character size for the sub-titles, used with par("cex")*csub
cpoi a character size for the points
cet a coefficient for the size of segments in standard deviation
... further arguments passed to or from other methods

Details

In a "reciprocal scaling", the reference score is a numeric code centred and normalized of the non zero cells of the array which both maximizes the variance of means by row and by column. The bars are drawn with half the length of this standard deviation.

Value

return a data.frame with the scores, weights and factors of correspondences (non zero cells)

Author(s)

Daniel Chessel

References

Examples

```r
layout(matrix(c(1,1,2,3), 2, 2), resp = FALSE)
data(nbfa)
d1 <- dudi.coa(nbfa, scan = FALSE)
score(d1, clab.r = 0, clab.c = 0.75)
head(recscal)
abline(v = 1, lty = 2, lwd = 3)
sco.distri(d1$l1[,1], nbfa)
sco.distri(d1$c1[,1], data.frame(t(nbfa)))
```

# 1 reciprocal scaling correspondence score -> species amplitude + sample diversity
# 2 sample score -> averaging -> species amplitude
# 3 species score -> averaging -> sample diversity

```r
layout(matrix(c(1,1,2,3), 2, 2), resp = FALSE)
data(rpjd1)
rpjdl1 <- dudi.coa(rpjd1$fau, scan = FALSE)
score(rpjdl1, clab.r = 0, clab.c = 0.75)
if (requireNamespace("MASS", quietly = TRUE)) {
  data(caith, package = "MASS")
score(dudi.coa(caith, scan = FALSE), clab.r = 1.5, clab.c = 1.5, cpoi = 3)
data(housetasks)
score(dudi.coa(housetasks, scan = FALSE), clab.r = 1.25, clab.c = 1.25,
  csub = 0, cpoi = 3)
}
par(mfrow = c(1,1))
score(rpjdl1, dotchart = TRUE, clab.r = 0)
```

score.mix

**Graphs to Analyse a factor in a Mixed Analysis**

**Description**

performs the canonical graph of a mixed analysis.

**Usage**

```r
## S3 method for class 'mix'
score(x, xax = 1, csub = 2, mfrow = NULL, which.var = NULL, ...)
```

**Arguments**

- `x` an object of class `mix`
- `xax` the column number for the used axis
- `csub` a character size for the sub-titles, used with `par("cex")*csub`
- `mfrow` a vector of the form "c(nr,nc)", otherwise computed by a special own function n2mfrow
which.var       the numbers of the kept columns for the analysis, otherwise all columns
...              further arguments passed to or from other methods

Author(s)

Daniel Chessel

Examples

data(lascaux)
w <- cbind.data.frame(lascaux$colo, lascaux$ornem)
dd <- dudi.mix(w, scan = FALSE, nf = 4, add = TRUE)
score(dd, which = which(dd$cr[,1] > 0.3))

Description

performs the canonical graph of a Principal Component Analysis.

Usage

## S3 method for class 'pca'
score(x, xax = 1, which.var = NULL, mfrow = NULL, csub = 2,
      sub = names(x$tab), abline = TRUE, ...)

Arguments

x            an object of class pca
xax          the column number for the used axis
which.var    the numbers of the kept columns for the analysis, otherwise all columns
mfrow        a vector of the form "c(nr,nc)", otherwise computed by a special own function
csub         a character size for sub-titles, used with par("cex")*csub
sub          a vector of string of characters to be inserted as sub-titles, otherwise the names
             of the variables
abline       a logical value indicating whether a regression line should be added
...          further arguments passed to or from other methods

Author(s)

Daniel Chessel
Examples

```r
data(deug)
   dd1 <- dudi.pca(deug$tab, scan = FALSE)
   score(dd1)
   
   # The correlations are :
   dd1$co[,1]
   # [1]  0.7925  0.6532  0.7410  0.5287  0.5539  0.7416  0.3336  0.2755  0.4172
```

---

**seconde**

**Students and Subjects**

Description

The seconde data frame gives the marks of 22 students for 8 subjects.

Usage

```r
data(seconde)
```

Format

This data frame (22,8) contains the following columns:
- H GEO: History and Geography
- FR AN: French literature
- PHYS: Physics
- MATH: Mathematics
- BIOL: Biology
- ECON: Economy
- ANGL: English language
- ESPA: Spanish language

Source

Personal communication

Examples

```r
data(seconde)
   if(adegraphicsLoaded()) {
     scatter(dudi.pca(seconde, scan = FALSE), row.plab.cex = 1, col.plab.cex = 1.5)
   } else {
     scatter(dudi.pca(seconde, scan = FALSE), clab.r = 1, clab.c = 1.5)
   }
```
Description

performs K separated multivariate analyses of an object of class ktab containing K tables.

Usage

sepan(X, nf = 2)
## S3 method for class 'sepan'
plot(x, mfrow = NULL, csub = 2, ...)
## S3 method for class 'sepan'
summary(object, ...)
## S3 method for class 'sepan'
print(x, ...)

Arguments

X an object of class ktab
nf an integer indicating the number of kept axes for each separated analysis
x, object an object of class 'sepan'
mfrow a vector of the form "c(nr,nc)", otherwise computed by a special own function n2mfrow
csub a character size for the sub-titles, used with par("cex")*csub
... further arguments passed to or from other methods

Details

The function plot on a sepan object allows to compare inertias and structures between arrays. In black, the eigenvalues of kept axes in the object 'sepan'.

Value

returns a list of class 'sepan' containing:
call a call order
tab.names a vector of characters with the names of tables
blo a numeric vector with the numbers of columns for each table
rank a numeric vector with the rank of the studied matrix for each table
Eig a numeric vector with all the eigenvalues
Li a data frame with the row coordinates
L1 a data frame with the row normed scores
Co a data frame with the column coordinates
The `skulls` data frame has 150 rows (Egyptian skulls) and 4 columns (anthropometric measures). The four variables are the maximum breadth (V1), the basibregmatic height (V2), the basialveolar length (V3) and the nasal height (V4). All measurements were taken in millimeters.

The measurements are made on 5 groups and 30 Egyptian skulls. The groups are defined as follows:

1 - the early predynastic period (circa 4000 BC)
2 - the late predynastic period (circa 3300 BC)
3 - the 12th and 13th dynasties (circa 1850 BC)
4 - the Ptolemiac period (circa 200 BC)
5 - the Roman period (circa 150 BC)
Source

References
The example is treated pp. 6, 13, 51, 64, 72, 107, 112 and 117.

Examples
data(skulls)
pca1 <- dudi.pca(skulls, scan = FALSE)
fac <- gl(5, 30)
levels(fac) <- c("-4000", "-3300", "-1850", "-200", "+150")
dis.skulls <- discrimin(pca1, fac, scan = FALSE)
if(!adegraphicsLoaded())
  plot(dis.skulls, 1, 1)

statico

STATIS and Co-Inertia : Analysis of a series of paired ecological tables

Description
Does the analysis of a series of pairs of ecological tables. This function uses Partial Triadic Analysis (pta) and ktab.match2ktabs to do the computations.

Usage
statico(KTX, KTY, scannf = TRUE)

Arguments
KTX an objet of class ktab
KTY an objet of class ktab
scannf a logical value indicating whether the eigenvalues bar plot should be displayed

Details
This function takes 2 ktabs and crosses each pair of tables of these ktabs with the function ktab.match2ktabs.
It then does a partial triadic analysis on this new ktab with pta.

Value
a list of class ktab, subclass kcoinertia. See ktab
WARNING

IMPORTANT: KTX and KTY must have the same k-tables structure, the same number of columns, and the same column weights.

Author(s)

Jean Thioulouse <jean.thioulouse@univ-lyon1.fr>

References


Examples

data(meau)
wit1 <- withinpca(meau$env, meau$design$season, scan = FALSE, scal = "total")
specpa <- dudi.pca(meau$spe, scale = FALSE, scan = FALSE, nf = 2)
wit2 <- wca(specpa, meau$design$season, scan = FALSE, nf = 2)
kt1 <- ktab.within(wit1, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kt2 <- ktab.within(wit2, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
statico1 <- statico(kt1, kt2, scan = FALSE)
plot(statico1)
kplot(statico1)

---

**statico.krandtest**  
*Monte-Carlo test on a Statico analysis (in C).*

Description

Performs the series of Monte-Carlo coinertia tests of a Statico analysis (one for each couple of tables).

Usage

```r
statico.krandtest(KTX, KTY, nrepet = 999, ...)
```

Arguments

- **KTX**: an objet of class ktab containing the environmental data
- **KTY**: an objet of class ktab containing the species data
- **nrepet**: the number of permutations
- ... further arguments passed to or from other methods
Details

This function takes 2 ktabs and does a coinertia analysis with coinertia on each pair of tables. It then uses the randtest function to do a permutation test on each of these coinertia analyses.

Value

krandtest, a list of randtest objects. See krandtest

WARNING

IMPORTANT: KTX and KTY must have the same k-tables structure, the same number of columns, and the same column weights.

Author(s)

Jean Thioulouse <jean.thioulouse@univ-lyon1.fr>

References


Examples

data(meau)
wit1 <- withinpca(meau$env, meau$design$season, scan = FALSE, scal = "total")
spepca <- dudi.pca(meau$spe, scale = FALSE, scan = FALSE, nf = 2)
wt2 <- wca(spepca, meau$design$season, scan = FALSE, nf = 2)
kta1 <- ktab.within(wit1, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
kta2 <- ktab.within(wit2, colnames = rep(c("S1","S2","S3","S4","S5","S6"), 4))
statico1 <- statico(kta1, kta2, scan = FALSE)
kr1 <- statico.krandtest(kta1, kta2)
plot(kr1)

statis

STATIS, a method for analysing K-tables

Description

performs a STATIS analysis of a ktab object.

Usage

statis(X, scannf = TRUE, nf = 3, tol = 1e-07)
## S3 method for class 'statis'
plot(x, xax = 1, yax = 2, option = 1:4, ...)
## S3 method for class 'statis'
print(x, ...)
Arguments

- **X**: an object of class 'ktab'
- **scannf**: a logical value indicating whether the number of kept axes for the compromise should be asked
- **nf**: if scannf FALSE, an integer indicating the number of kept axes for the compromise
- **tol**: a tolerance threshold to test whether the distance matrix is Euclidean: an eigenvalue is considered positive if it is larger than \(-tol*\lambda_1\) where \(\lambda_1\) is the largest eigenvalue
- **x**: an object of class 'statis'
- **xax, yax**: the numbers of the x-axis and the y-axis
- **option**: an integer between 1 and 4, otherwise the 4 components of the plot are displayed
- **...**: further arguments passed to or from other methods

Value

`statis` returns a list of class 'statis' containing:

- **RV**: a matrix with the all RV coefficients
- **RV.eig**: a numeric vector with all the eigenvalues
- **RV.coo**: a data frame with the array scores
- **tab.names**: a vector of characters with the names of the arrays
- **RV.tabw**: a numeric vector with the array weigths
- **C.nf**: an integer indicating the number of kept axes
- **C.rank**: an integer indicating the rank of the analysis
- **C.li**: a data frame with the row coordinates
- **C.Co**: a data frame with the column coordinates
- **C.T4**: a data frame with the principal vectors (for each table)
- **TL**: a data frame with the factors (not used)
- **TC**: a data frame with the factors for Co
- **T4**: a data frame with the factors for T4

Author(s)

Daniel Chessel

References


Examples

```r
data(jv73)
kt1 <- ktab.within(withinpca(jv73$morpho, jv73$fac.riv, scann = FALSE))
statis1 <- statis(kt1, scann = FALSE)
plot(statis1)

dudi1 <- dudi.pca(jv73$poi, scann = FALSE, scal = FALSE)
wit1 <- wca(dudi1, jv73$fac.riv, scann = FALSE)
kt3 <- ktab.within(wit1)
data(jv73)
statis3 <- statis(kt3, scann = FALSE)
plot(statis3)

if(adegraphicsLoaded()) {
  s.arrow(statis3$C.li, pgrid.text.cex = 0)
  kplot(statis3, traj = TRUE, arrow = FALSE, plab.cex = 0, psub.cex = 3, ppoi.cex = 3)
} else {
  s.arrow(statis3$C.li, cgrid = 0)
  kplot(statis3, traj = TRUE, arrow = FALSE, unique = TRUE,
        clab = 0, csub = 3, cpoi = 3)
}

statis3
```

---

steppe

*Transect in the Vegetation*

---

Description

This data set gives the presence-absence of 37 species on 515 sites.

Usage

```r
data(steppe)
```

Format

steppe is a list of 2 components.

- **tab** is a data frame with 512 rows (sites) and 37 variables (species) in presence-absence.
- **esp.names** is a vector of the species names.

Source

Examples

```r
par(mfrow = c(3,1))
data(steppe)
w1 <- col(as.matrix(steppe$tab[,1:15]))
w1 <- as.numeric(w1[steppe$tab[,1:15] > 0])
w2 <- row(as.matrix(steppe$tab[,1:15]))
w2 <- as.numeric(w2[steppe$tab[,1:15] > 0])
plot(w2, w1, pch = 20)
plot(dudi.pca(steppe$tab, scan = FALSE, scale = FALSE)$li[,1],
     pch = 20, ylab = "PCA", xlab = ",", type = "b")
plot(dudi.coa(steppe$tab, scan = FALSE)$li[,1], pch = 20,
     ylab = "COA", xlab = ",", type = "b")
par(mfrow = c(1,1))
```

supcol                Projections of Supplementary Columns

Description

performs projections of supplementary columns.

Usage

```r
supcol(x, ...)  # S3 method for class 'dudi'
supcol(x, Xsup, ...)  # S3 method for class 'coa'
```

Arguments

- **x**: an object used to select a method
- **Xsup**: an array with the supplementary columns (Xsup and x$tab have the same row number)
- **...**: further arguments passed to or from other methods

Details

If supcol.dudi is used, the column vectors of Xsup are projected without prior modification onto the principal components of dudi with the scalar product associated to the row weightings of dudi.

Value

A list of two components:

- **tabsup**: data frame containing the array with the supplementary columns transformed or not
- **cosup**: data frame containing the coordinates of the supplementary projections
**supdist**

Projection of additional items in a PCO analysis

**Description**

This function takes the grand distance matrix between all items (Active + Supplementary). It computes the PCO of the distance matrix between Active items, and projects the distance matrix of Supplementary items in this PCO.

**Usage**

```r
supdist(d, fsup, tol = 1e-07)
```

**Arguments**

- `d`: Grand distance matrix between all (Active + Supplementary) items
- `fsup`: A factor with two levels giving the Active (level ‘A’) or Supplementary (level ‘S’) status for each item in the distance matrix.
- `tol`: Numeric tolerance used to evaluate zero eigenvalues

**Value**

- `coordSup`: Coordinates of Supplementary items projected in the PCO of Active items
- `coordAct`: Coordinates of Active item
- `coordTot`: Coordinates of Active plus Supplementary items
Author(s)
Jean Thioulouse

References

See Also
dudi.pco, suprow

Examples
```r
data(meau)
## Case 1: Supplementary items = subset of Active items
## Supplementary coordinates should be equal to Active coordinates
## PCO of active items (meau dataset has 6 sites and 10 variables)
envpca1 <- dudi.pca(meau$env, scannf = FALSE)
dAct <- dist(envpca1$tab)
pc1 <- dudi.pco(dAct, scannf = FALSE)
## Projection of rows 19:24 (winter season for the 6 sites)
## Supplementary items must be normalized
f1 <- function(w) (w - envpca1$cent) / envpca1$norm
envSup <- t(apply(meau$env[19:24, ], 1, f1))
envTot <- rbind.data.frame(envpca1$tab, envSup)
dTot <- dist(envTot)
fsa1 <- as.factor(rep(c("A", "S"), c(24, 6)))
cSup1 <- supdist(dTot, fsa1)
## Comparison (coordinates should be equal)
cSup1$coordSup[, 1:2]
pco1$li[19:24, ]
```
```r
data(meaudret)
## Case 2: Supplementary items = new items
## PCO of active items (meaudret dataset has only 5 sites and 9 variables)
envpca2 <- dudi.pca(meaudret$env, scannf = FALSE)
dAct <- dist(envpca2$tab)
pc2 <- dudi.pco(dAct, scannf = FALSE)
## Projection of site 6 (four seasons, without Oxyg variable)
## Supplementary items must be normalized
f1 <- function(w) (w - envpca2$cent) / envpca2$norm
envSup <- t(apply(meaudret$env[seq(6, 24, 6), -5], 1, f1))
envTot <- rbind.data.frame(envpca2$tab, envSup)
dTot <- dist(envTot)
fsa2 <- as.factor(rep(c("A", "S"), c(20, 4)))
cSup2 <- supdist(dTot, fsa2)
## Supplementary items vs. real items
if(!aderaphicsLoaded()) {
  par(mfrow = c(1, 2))
s.label(pco1$li)
```
suprow

Projections of Supplementary Rows

Description

This function performs a projection of supplementary rows (i.e. supplementary individuals).

Usage

```r
## S3 method for class 'coa'
suprow(x, Xsup, ...)
## S3 method for class 'dudi'
suprow(x, Xsup, ...)
## S3 method for class 'dudi'
predict(object, newdata, ...)
## S3 method for class 'pca'
suprow(x, Xsup, ...)
## S3 method for class 'acm'
suprow(x, Xsup, ...)
## S3 method for class 'mix'
suprow(x, Xsup, ...)
## S3 method for class 'fca'
suprow(x, Xsup, ...)
```

Arguments

- `x`, `object` an object of class `dudi`
- `Xsup`, `newdata` an array with the supplementary rows
- `...` further arguments passed to or from other methods

Details

If `suprow.dudi` is used, the column vectors of `Xsup` are projected without prior modifications onto the principal components of `dudi` with the scalar product associated to the row weightings of `dudi`.

Value

`predict` returns a data frame containing the coordinates of the supplementary rows. `suprow` returns a list with the transformed table `Xsup` in `tabsup` and the coordinates of the supplementary rows in `lisup`.
Author(s)
Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References

Examples
```r
data(euro123)
par(mfrow = c(2, 2))
w <- euro123[[2]]
dudi1 <- dudi.pca(w, scal = FALSE, scan = FALSE)
if(adegraphicsLoaded()) {
  g11 <- s.arrow(dudi1$c1, psub.text = "Classical", psub.posi = "bottomright", plot = FALSE)
  g12 <- s.label(suprow(dudi1, w)$tabsup, plab.cex = 0.75, plot = FALSE)
  g1 <- superpose(g11, g12)
  g21 <- s.arrow(dudi1$c1, psub.text = "Without centring", psub.posi = "bottomright", plot = FALSE)
  g22 <- s.label(suprow(dudi1, w)$tabsup, plab.cex = 0.75, plot = FALSE)
  g2 <- superpose(g21, g22)
  g3 <- triangle.label(w, plab.cex = 0.75, label = row.names(w), adjust = FALSE, plot = FALSE)
  g4 <- triangle.label(w, plab.cex = 0.75, label = row.names(w), adjust = TRUE, plot = FALSE)
  G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  s.arrow(dudi1$c1, sub = "Classical", possub = "bottomright", csub = 2.5)
  s.label(suprow(dudi1, w), add.plot = TRUE, clab = 0.75)
  s.arrow(dudi1$c1, sub = "Without centring", possub = "bottomright", csub = 2.5)
  s.label(suprow(dudi1, w), add.plot = TRUE)
  triangle.plot(w, clab = 0.75, label = row.names(w), scal = FALSE)
  triangle.plot(w, clab = 0.75, label = row.names(w), scal = TRUE)
}
data(rpjdl)
rpjdl.coa <- dudi.coa(rpjdl$fau, scann = FALSE, nf = 4)
rpjdl.coa$li[1:3, ]
suprow(rpjdl.coa, rpjdl$fau[1:3, ])$lisup #the same

data(deug)
deug.dudi <- dudi.pca(df = deug$tab, center = deug$cent, scale = FALSE, scannf = FALSE)
suprow(deug.dudi, deug$tab[1:3, ])$lisup #the supplementary individuals are centered
deug.dudi$li[1:3, ] # the same
```
Description

This function performs a projection of supplementary rows (i.e. supplementary individuals) for a Partial Triadic Analysis (pta) of K-tables. Computations are valid ONLY if the pta has been done on a K-Tables obtained by the withinpca function, followed by calls to the ktab.within and t functions.

Usage

## S3 method for class 'pta'
suprow(x, Xsup, facSup, ...)

Arguments

- x: an object of class pta
- Xsup: a table with the supplementary rows
- facSup: a factor partitioning the rows of Xsup
- ...: further arguments passed to or from other methods

Details

This function computes the coordinates of the supplementary rows for a K-tables. The table of supplementary rows is standardized according to the 'Bouroche' standardization used in the Within Analysis of the original pta. In a first step, the table of supplementary rows is standardized (centred and normed) with the mean and variance of the original table of active individuals (i.e. the K-tables used in pta). Then, according to the withinpca procedure, a second transformation is applied.

For "partial", supplementary rows are standardized in each sub-table (corresponding to each level of the factor) by the mean and variance of each corresponding sub-sample in the table of active individuals. Hence, supplementary rows have null mean and unit variance in each sub-table.

For "total", supplementary rows are centred in each sub-table with the mean of each corresponding sub-sample in the table of active individuals and then normed with the global variance of the table of active individuals. Hence, supplementary rows have a null mean in each sub-table and a global variance equal to one.

Value

Returns a list with the transformed table Xsup in tabsup and the coordinates of the supplementary rows in lisup.

Author(s)

Benjamin Alric <benjamin.alric@irstea.fr>
Jean Thioulouse <jean.thioulouse@univ-lyon1.fr>
References


Examples

data(meau)
# Active rows
actenv <- meau$env[meau$design$site != "S6", -c(5)]
actfac <- meau$design$season[meau$design$site != "S6"]
# Supplementary rows
supenv <- meau$env[meau$design$site == "S6", -c(5)]
supfac <- meau$design$season[meau$design$site == "S6"]
# Total = active + supplementary rows
totenv <- meau$env[, -c(5)]
totfac <- meau$design$season
# PTA with 6 sampling sites
wittot <- withinpca(df = totenv, fac = totfac, scannf = FALSE, scaling = "partial")
ktaltot <- ktab.within(wittot, colnames = rep(c("S1", "S2", "S3", "S4", "S5", "S6"), 4))
ktatot <- t(ktaltot)
pta1tot <- pta(ktatot, scann = FALSE)
# PTA with 5 sampling sites and site 6 added as supplementary element
wit1 <- withinpca(df = actenv, fac = actfac, scannf = FALSE, scaling = "partial")
kt1 <- ktab.within(wit1, colnames = rep(c("S1", "S2", "S3", "S4", "S5"), 4))
kt2 <- t(kt1)
pta1 <- pta(kt2, scann = FALSE)
supenv.pta <- suprow(x = pta1, Xsup = supenv, facSup = supfac)
if (adegraphicsLoaded()) {
  # g1t = active + supplementary rows
  g1t <- s.label(pta1tot$Tli, labels = rownames(totenv),
                 plabels = list(box = list(draw = FALSE), optim = TRUE),
                 xlim = c(-6, 5), ylim = c(-5, 5),
                 psub = list(text="Total", position="topleft"),
                 plot = FALSE)
  # g1 = Active rows
  g1 <- s.label(pta1$Tli, labels = rownames(actenv),
                plabels = list(box = list(draw = FALSE), optim = TRUE),
                xlim = c(-6, 5), ylim = c(-5, 5),
                psub = list(text="Active", position="topleft"),
                pgrid = list(text=list(cex = 0)),
                plot = FALSE)
  # g2 = Supplementary rows
  g2 <- s.label(supenv.pta$lisup, plabels = list(box = list(draw = FALSE), optim = TRUE),
                ppoints = list(col = "red"),
                psub = list(text="Supplementary", position="topright"),
                pgrid = list(text=list(cex = 0)),
                plot = FALSE)
  # g3 = superposition of active and supplementary rows
  g3 <- g1 + g2
  # Comparison of the total analysis and the analysis with supplementary rows
  ADEgS(list(g1t,g3))
} else {
  par(mfrow=c(2,2))
  # g1t = active + supplementary rows
  g1t <- s.label(pta1tot$Tli, label = rownames(totenv), xlim = c(-6, 5), ylim = c(-5, 5),
                 sub="Total")
  # g1 = Active rows
  g1 <- s.label(pta1$Tli, label = rownames(actenv), clabel = 1, xlim = c(-6, 5),
                 sub="Active")
  # g2 = Supplementary rows
  g2 <- s.label(supenv.pta$lisup, ppoints = list(col = "red"),
                 psub = list(text="Supplementary", position="topright"),
                 pgrid = list(text=list(cex = 0)),
                 plot = FALSE)
  # g3 = superposition of active and supplementary rows
  g3 <- g1 + g2
  # Comparison of the total analysis and the analysis with supplementary rows
  ADEgS(list(g1t,g3))
}
symbols.phylog

```r
ylim = c(-5, 5), sub="Active+Supplementary")
# g2 = Supplementary rows
g2 <- s.label(supenv.pta$lisup, clabel = 1.5, xlim = c(-6, 5), ylim = c(-5, 5),
  add.plot = TRUE)
```

**symbols.phylog**  
*Representation of a quantitative variable in front of a phylogenetic tree*

### Description

symbols.phylog draws the phylogenetic tree and represents the values of the variable by symbols (squares or circles) which size is proportional to value. White symbols correspond to values which are below the mean, and black symbols correspond to values which are over.

### Usage

```r
symbols.phylog(phylog, circles, squares, csize = 1, clegend = 1,
  sub = "", csub = 1, possub = "topleft")
```

### Arguments

- **phylog**: an object of class phylog  
- **circles**: a vector giving the radii of the circles  
- **squares**: a vector giving the length of the sides of the squares  
- **csize**: a size coefficient for symbols  
- **clegend**: a character size for the legend used by `par("cex")*clegend`  
- **sub**: a string of characters to be inserted as legend  
- **csub**: a character size for the legend, used with `par("cex")*csub`  
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")

### Author(s)

Daniel Chessel  
Sébastien Ollier <sebastien.ollier@u-psud.fr>

### See Also

`table.phylog` and `dotchart.phylog` for many variables
Examples

data(mjrochet)
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab0 <- data.frame(scalewt(log(mjrochet$tab)))
par(mfrow=c(3,2))
for (j in 1:6) {
  w <- tab0[,j]
  symbols.phylog(phylog = mjrochet.phy, w, csi = 1.5, cleg = 1.5,
                 sub = names(tab0)[j], csub = 3)
}
par(mfrow=c(1,1))

syndicats

Two Questions asked on a Sample of 1000 Respondents

Description

This data set is extracted from an opinion poll (period 1970-1980) on 1000 respondents.

Usage

data(syndicats)

Format

The syndicats data frame has 5 rows and 4 columns.
"Which politic family are you agreeing about?" has 5 response items: extgauche (extreme left)
left center right and extdroite (extreme right)
"What do you think of the trade importance?" has 4 response items: trop (too important) adequate
insufficient nesaispas (no opinion)

Source

unknown

Examples

data(syndicats)
par(mfrow = c(1,2))
dud1 <- dudi.coa(syndicats, scan = FALSE)
score (dud1, 1, TRUE)
score (dud1, 1, FALSE)
Average temperatures of 30 French cities

Description
This data set gives the average temperatures of 30 French cities during 12 months.

Usage
data(t3012)

Format
t3012 is a list with the following components:

xy a data frame with 30 rows (cities) and 2 coordinates (x, y)

temp a data frame with 30 rows (cities) and 12 columns (months). Each column contains the average temperature in tenth of degree Celsius.

contour a data frame with 4 columns (x1, y1, x2, y2) for the contour display of France

Spatial an object of the class SpatialPolygons of sp, containing the map

Source

Examples
data(t3012)
data(elec88)

if(adegraphicsLoaded()) {
  if(requireNamespace("sp", quietly = TRUE)) {
    s.arrow(t3012$xy, pori.ori = as.numeric(t3012$xy["Paris", ]), Sp = t3012$Spatial, pSp.col = "white", pgrid.draw = FALSE)
  }
} else {
  area.plot(elec88$area)
  s.arrow(t3012$xy, ori = as.numeric(t3012$xy["Paris", ]), add.p = TRUE)
}
Plot of Contingency Tables

Description

presents a graph for viewing contingency tables.

Usage

table.cont(df, x = 1:ncol(df), y = 1:nrow(df),
            row.labels = row.names(df), col.labels = names(df),
            clabel.row = 1, clabel.col = 1, abmean.x = FALSE, abline.x = FALSE,
            abmean.y = FALSE, abline.y = FALSE, csize = 1, clegend = 0, grid = TRUE)

Arguments

df a data frame with only positive or null values
x a vector of values to position the columns
y a vector of values to position the rows
row.labels a character vector for the row labels
col.labels a character vector for the column labels
clabel.row a character size for the row labels
clabel.col a character size for the column labels
abmean.x a logical value indicating whether the column conditional means should be drawn
abline.x a logical value indicating whether the regression line of y onto x should be plotted
abmean.y a logical value indicating whether the row conditional means should be drawn
abline.y a logical value indicating whether the regression line of x onto y should be plotted
csize a coefficient for the square size of the values
clegend if not NULL, a character size for the legend used with par("cex")*clegend
grid a logical value indicating whether a grid in the background of the plot should be drawn

Author(s)

Daniel Chessel
Examples

```r
data(chats)
chatsw <- data.frame(t(chats))
chatscoa <- dudi.coa(chatsw, scann = FALSE)
par(mfrow = c(2,2))
table.cont(chatsw, abmean.x = TRUE, csi = 2, abline.x = TRUE, 
clabel.r = 1.5, clabel.c = 1.5)
table.cont(chatsw, abmean.y = TRUE, csi = 2, abline.y = TRUE, 
clabel.r = 1.5, clabel.c = 1.5)
table.cont(chatsw, x = chatscoa$c1[,1], y = chatscoa$l1[,1], 
    abmean.x = TRUE, csi = 2, abline.x = TRUE, clabel.r = 1.5, clabel.c = 1.5)
table.cont(chatsw, x = chatscoa$c1[,1], y = chatscoa$l1[,1], 
    abmean.y = TRUE, csi = 2, abline.y = TRUE, clabel.r = 1.5, clabel.c = 1.5)
par(mfrow = c(1,1))

## Not run:
data(rpjdl)
w <- data.frame(t(rpjdl$fau))
wcoa <- dudi.coa(w, scann = FALSE)
table.cont(w, abmean.y = TRUE, x = wcoa$c1[,1], y = rank(wcoa$l1[,1]), 
    csi = 0.2, clabel.c = 0, row.labels = rpjdl$lalab, clabel.r = 0.75)
## End(Not run)
```

---

**table.dist**  
*Graph Display for Distance Matrices*

**Description**

presents a graph for viewing distance matrices.

**Usage**

```r
table.dist(d, x = 1:(attr(d, "Size")), labels = as.character(x), 
    clabel = 1, csize = 1, grid = TRUE)
```

**Arguments**

- `d`  
an object of class dist
- `x`  
a vector of the row and column positions
- `labels`  
a vector of strings of characters for the labels
- `clabel`  
a character size for the labels
- `csize`  
a coefficient for the circle size
- `grid`  
a logical value indicating whether a grid in the background of the plot should be drawn
Author(s)
Daniel Chessel

Examples
data(eurodist)
table.dist(eurodist, labels = attr(eurodist, "Labels"))

---

### table.paint

**Plot of the arrays by grey levels**

#### Description

presents a graph for viewing the numbers of a table by grey levels.

#### Usage

```r
table.paint(df, x = 1:ncol(df), y = nrow(df):1,
            row.labels = row.names(df), col.labels = names(df),
            clabel.row = 1, clabel.col = 1, csize = 1, clegend = 1)
```

#### Arguments

- **df**: a data frame
- **x**: a vector of values to position the columns, used only for the ordered values
- **y**: a vector of values to position the rows, used only for the ordered values
- **row.labels**: a character vector for the row labels
- **col.labels**: a character vector for the column labels
- **clabel.row**: a character size for the row labels
- **clabel.col**: a character size for the column labels
- **csize**: if 'clegend' not NULL, a coefficient for the legend size
- **clegend**: a character size for the legend, otherwise no legend

#### Author(s)
Daniel Chessel
Examples

```r
data(rpjdl)
X <- data.frame(t(rpjdl$fau))
Y <- data.frame(t(rpjdl$mil))
layout(matrix(c(1,2,2,2,2,2,1,2,2,1,2,2,2,1,2,2,2,4, 4, 4)), 4)
coa1 <- dudi.coa(X, scan = FALSE)
x <- rank(coa1$co[,1])
y <- rank(coa1$li[,1])
table.paint(Y, x = x, y = 1:8, clabel.c = 0, cleg = 0)
abline(v = 114.9, lwd = 3, col = "red")
abline(v = 66.4, lwd = 3, col = "red")
table.paint(X, x = x, y = y, clabel.c = 0, cleg = 0,
           row.lab = paste(" ", row.names(X), sep = ""))
abline(v = 114.9, lwd = 3, col = "red")
abline(v = 66.4, lwd = 3, col = "red")
par(mfrow = c(1, 1))
```

**table.phylog**

*Plot arrays in front of a phylogenetic tree*

Description

This function gives a graphical display for viewing the numbers of a table by square sizes in front of the corresponding phylogenetic tree.

Usage

```r
table.phylog(df, phylog, x = 1:ncol(df), f.phylog = 0.5,
            labels.row = gsub("[\_]", " ", row.names(df)), clabel.row = 1,
            labels.col = names(df), clabel.col = 1,
            labels.nod = names(phylog$nodes), clabel.nod = 0, cleaves = 1,
            cnodes = 1, csize = 1, grid = TRUE, clegend = 0.75)
```

Arguments

- `df`: a data frame or a matrix
- `phylog`: an object of class 'phylog'
- `x`: a vector of values to position the columns
- `f.phylog`: a size coefficient for tree size (a parameter to draw the tree in proportion to leaves labels)
- `labels.row`: a vector of strings of characters for row labels
- `clabel.row`: a character size for the leaves labels, used with `par("cex")*clabel.row`. If zero, no row labels are drawn
- `labels.col`: a vector of strings of characters for columns labels
- `clabel.col`: a character size for the leaves labels, used with `par("cex")*clabel.col`. If zero, no column labels are drawn
table.value

labels.nod : a vector of strings of characters for the nodes labels
clabel.nod : a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
cleaves : a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
cnodes : a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
csize : a size coefficient for symbols
grid : a logical value indicating whether the grid should be plotted
clegend : a character size for the legend (if 0, no legend)

Details

The function verifies that sort(row.names(df))==sort(names(phylog$leaves)). If df is a matrix the function uses as.data.frame(df).

Author(s)

Daniel Chessel
Sébastien Ollier <sebastien.ollier@u-psud.fr>

See Also

symbols.phylog for one variable

Examples

```r
## Not run:
data(newick.eg)
w.phy <- newick2phylog(newick.eg[[9]])
w.tab <- data.frame(matrix(rnorm(620), 31, 20))
row.names(w.tab) <- sort(names(w.phy$leaves))
table.phylog(w.tab, w.phy, csi = 1.5, f = 0.5, clabel.n = 0.75, clabel.c = 0.5)
## End(Not run)
```

Description

presents a graph for viewing the numbers of a table by square sizes.
table.value

Usage

    table.value(df, x = 1:ncol(df), y = nrow(df):1,
               row.labels = row.names(df), col.labels = names(df), clabel.row = 1,
               clabel.col = 1, csize = 1, clegend = 1, grid = TRUE)

Arguments

    df             a data frame
    x              a vector of values to position the columns
    y              a vector of values to position the rows
    row.labels     a character vector for the row labels
    col.labels     a character vector for the column labels
    clabel.row     a character size for the row labels
    clabel.col     a character size for the column labels
    csize          a coefficient for the square size of the values
    clegend        a character size for the legend (if 0, no legend)
    grid           a logical value indicating whether the grid should be plotted

Author(s)

    Daniel Chessel

Examples

    if(!adegraphicsLoaded()) {
      data(olympic)
      w <- olympic$tab
      w <- data.frame(scale(w))
      wpca <- dudi.pca(w, scann = FALSE)
      par(mfrow = c(1, 3))
      table.value(w, csi = 2, clabel.r = 2, clabel.c = 2)
      table.value(w, y = rank(wpca$li[, 1]), x = rank(wpca$co[, 1]), csi = 2,
                   clabel.r = 2, clabel.c = 2)
      table.value(w, y = wpca$li[, 1], x = wpca$co[, 1], csi = 2,
                   clabel.r = 2, clabel.c = 2)
      par(mfrow = c(1, 1))
    }
Description

This data set gives informations between sites, species, environmental and biological variables.

Usage

data(tarentaise)

Format

tarentaise is a list of 5 components.

edcol is a data frame with 376 sites and 98 bird species.

frnames is a vector of the 98 French names of the species.

alti is a vector giving the altitude of the 376 sites in m.

envir is a data frame with 14 environmental variables.

traits is a data frame with 29 biological variables of the 98 species.

Details

The attribute col.blocks of the data frame tarentaise$traits indicates it is composed of 6 units of variables.

Source

Original data from Hubert Tournier, University of Savoie and Philippe Lebreton, University of Lyon 1.

References


Examples

data(tarentaise)
coa1 <- dudi.coa(tarentaise$ecol, sca = FALSE, nf = 2)
s.class(coa1$li, tarentaise$envir$alti, wt = coa1$lw)
## Not run:
acm1 <- dudi.acm(tarentaise$envir, sca = FALSE, nf = 2)
s.class(acm1$li, tarentaise$envir$alti)
## End(Not run)

Description

This data sets contains two taxonomies.

Usage

data(taxo.eg)

Format

taxo.eg is a list containing the 2 following objects:

taxo.eg[[1]] is a data frame with 15 species and 3 columns.
taxo.eg[[2]] is a data frame with 40 species and 2 columns.

Details

Variables of the first data frame are : genre (a factor genre with 8 levels), famille (a factor familiy with 5 levels) and ordre (a factor order with 2 levels).

Variables of the second data frame are : gen(a factor genre with 29 levels), fam (a factor family with 19 levels).

Examples

data(taxo.eg)
taxo.eg[[1]]
as.taxo(taxo.eg[[1]])
class(taxo.eg[[1]])
class(as.taxo(taxo.eg[[1]]))
tax.phy <- taxo2phylog(as.taxo(taxo.eg[[1]])), add.tools = TRUE)
plot(tax.phy,clabel.l=1)
par(mfrow = c(1,2))
testdim  

Function to perform a test of dimensionality

Description

This functions allow to test for the number of axes in multivariate analysis. The procedure testdim.pca implements a method for principal component analysis on correlation matrix. The procedure is based on the computation of the RV coefficient.

Usage

testdim(object, ...)  
## S3 method for class 'pca'  
testdim(object, nrepet = 99, nbax = object$rank, alpha = 0.05, ...)

Arguments

object an object corresponding to an analysis (e.g. duality diagram, an object of class dudi)  
nrepet the number of repetitions for the permutation procedure  
nbax the number of axes to be tested, by default all axes  
alpha the significance level  
... other arguments

Value

An object of the class krandtest. It contains also:

nb The estimated number of axes to keep  
nb.cor The number of axes to keep estimated using a sequential Bonferroni procedure

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>

References

testdim.multiblock

See Also
dudi.pca, RV.rtest, testdim.multiblock

Examples

```r
tab <- data.frame(matrix(rnorm(200),20,10))
pca1 <- dudi.pca(tab, scannf=FALSE)
test1 <- testdim(pca1)
test1

test1$nb

test1$nb.cor

data(doubs)
pca2 <- dudi.pca(doubs$env, scannf=FALSE)
test2 <- testdim(pca2)
test2

test2$nb

test2$nb.cor
```

testdim.multiblock  Selection of the number of dimension by two-fold cross-validation for multiblock methods

Description

Function to perform a two-fold cross-validation to select the optimal number of dimensions of multiblock methods, i.e., multiblock principal component analysis with instrumental Variables or multiblock partial least squares

Usage

```r
## S3 method for class 'multiblock'

## S3 method for class 'multiblock'
testdim(object, nrepet = 100, quantiles = c(0.25, 0.75), ...)
```

Arguments

object  an object of class multiblock created by mbpls or mbpcaiv
nrepet  integer indicating the number of repetitions
quantiles a vector indicating the lower and upper quantiles to compute
... other arguments to be passed to methods

Value

An object of class krandxval

Author(s)

Stéphanie Bougeard (<stephanie.bougeard@anses.fr>) and Stéphane Dray (<stephane.dray@univ-lyon1.fr>)
References


See Also

mbpcaiv, mbpls, randboot.multiblock, as.krandxval

Examples

data(chickenk)
Mortality <- chickenk[[1]]
dudiY.chick <- dudi.pca(Mortality, center = TRUE, scale = TRUE, scannf = FALSE)
ktabX.chick <- ktab.list.df(chickenk[2:5])
resmbpcaiv.chick <- mbpcaiv(dudiY.chick, ktabX.chick, scale = TRUE,
option = "uniform", scannf = FALSE)
## nrepet should be higher for a real analysis
test <- testdim(resmbpcaiv.chick, nrepet = 10)
test
if(adegraphicsLoaded())
plot(test)

---

tintoodiel

**Tinto and Odiel estuary geochemistry**

Description

This data set contains informations about geochemical characteristics of heavy metal pollution in surface sediments of the Tinto and Odiel river estuary (south-western Spain).

Usage

data(tintoodiel)

Format

tintoodiel is a list with the following components:

- **xy** a data frame that contains spatial coordinates of the 52 sites
- **tab** a data frame with 12 columns (concentration of heavy metals) and 52 rows (sites)
- **neig** an object of class neig
- **nb** the neighbourhood graph of the 52 sites (an object of class nb)
tithonia

Phylogeny and quantitative traits of flowers

Description

This data set describes the phylogeny of 11 flowers as reported by Morales (2000). It also gives morphologic and demographic traits corresponding to these 11 species.

Usage

data(tithonia)

Format

tithonia is a list containing the 2 following objects:

tre is a character string giving the phylogenetic tree in Newick format.
tab is a data frame with 11 species and 14 traits (6 morphologic traits and 8 demographic).
Details

Variables of `tithonia$tab` are the following ones:

- `morho1`: is a numeric vector that describes the seed size (mm)
- `morho2`: is a numeric vector that describes the flower size (mm)
- `morho3`: is a numeric vector that describes the female leaf size (cm)
- `morho4`: is a numeric vector that describes the head size (mm)
- `morho5`: is a integer vector that describes the number of flowers per head
- `morho6`: is a integer vector that describes the number of seeds per head
- `demo7`: is a numeric vector that describes the seedling height (cm)
- `demo8`: is a numeric vector that describes the growth rate (cm/day)
- `demo9`: is a numeric vector that describes the germination time
- `demo10`: is a numeric vector that describes the establishment (per cent)
- `demo11`: is a numeric vector that describes the viability (per cent)
- `demo12`: is a numeric vector that describes the germination (per cent)
- `demo13`: is a integer vector that describes the resource allocation
- `demo14`: is a numeric vector that describes the adult height (m)

Source


Examples

```r
data(tithonia)
phy <- newick2phylog(tithonia$tre)
tab <- log(tithonia$tab + 1)
table.phylol(scalewt(tab), phy)
geary.moran(phy$Wmat, tab)
geary.moran(phy$Amat, tab)
```

---

### tortues

*Morphological Study of the Painted Turtle*

**Description**

This data set gives a morphological description (4 characters) of 48 turtles.

**Usage**

```r
data(tortues)
```

**Format**

A data frame with 48 rows and 4 columns (length (mm), maximum width(mm), height (mm), gender).
Source


Examples

```r
data(tortues)
xyz <- as.matrix(tortues[, 1:3])
ref <- -svd(xyz)$u[, 1]
pch0 <- c(1, 20)[as.numeric(tortues$sexe)]
plot(ref, xyz[, 1], ylim = c(40, 180), pch = pch0)
abline(lm(xyz[, 1] ~ -1 + ref))
points(ref,xyz[, 2], pch = pch0)
abline(lm(xyz[, 2] ~ -1 + ref))
points(ref,xyz[, 3], pch = pch0)
abline(lm(xyz[, 3] ~ -1 + ref))```

---

toxicity  

*Homogeneous Table*

Description

This data set gives the toxicity of 7 molecules on 17 targets expressed in \(-\log(\text{mol/liter})\)

Usage

```r
data(toxicity)```

Format

toxicity is a list of 3 components.

- **tab** is a data frame with 7 columns and 17 rows
- **species** is a vector of the names of the species in the 17 targets
- **chemicals** is a vector of the names of the 7 molecules

Source

Examples

data(toxicity)
if(adegraphicsLoaded()) {
  table.image(toxicity$tab, labelsy = toxicity$species, labelsx = toxicity$chemicals, nclass = 7,
              ptable.margin = list(b = 5, l = 25, t = 25, r = 5), ptable.y.pos = "left", pgrid.draw = TRUE)
  table.value(toxicity$tab, labelsy = toxicity$species, labelsx = toxicity$chemicals,
              ptable.margin = list(b = 5, l = 5, t = 25, r = 26))
} else {
  table.paint(toxicity$tab, row.lab = toxicity$species, col.lab = toxicity$chemicals)
  table.value(toxicity$tab, row.lab = toxicity$species, col.lab = toxicity$chemicals)
}

triangle.class

Triangular Representation and Groups of points

Description

Function to plot triangular data (i.e. dataframe with 3 columns of positive or null values) and a partition

Usage

triangle.class(ta, fac, col = rep(1, length(levels(fac))),
               wt = rep(1, length(fac)), cstar = 1, cellipse = 0, axesell = TRUE,
               label = levels(fac), clabel = 1, cpoint = 1, pch = 20, draw.line = TRUE,
               addaxes = FALSE, addmean = FALSE, labeltriangle = TRUE, sub = 
               possub = "bottomright", show.position = TRUE, scale = TRUE, min3 = NULL,
               max3 = NULL)

Arguments

- ta: a data frame with 3 columns of null or positive numbers
- fac: a factor of length the row number of ta
- col: a vector of color for showing the groups
- wt: a vector of row weighting for the computation of the gravity centers by class
- cstar: a character size for plotting the stars between 0 (no stars) and 1 (complete star) for a line linking a point to the gravity center of its belonging class.
- cellipse: a positive coefficient for the inertia ellipse size
- axesell: a logical value indicating whether the ellipse axes should be drawn
- label: a vector of strings of characters for the labels of gravity centers
- clabel: if not NULL, a character size for the labels, used with par("cex")*clabel
- cpoint: a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
triangle.class

- **pch**: if `cpoint > 0`, an integer specifying the symbol or the single character to be used in plotting points.
- **draw.line**: a logical value indicating whether the triangular lines should be drawn.
- **addaxes**: a logical value indicating whether the axes should be plotted.
- **addmean**: a logical value indicating whether the mean point should be plotted.
- **labeltriangle**: a logical value indicating whether the variable labels of `ta` should be drawn on the triangular sides.
- **sub**: a string of characters for the graph title.
- **csub**: a character size for plotting the graph title.
- **possub**: a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright").
- **show.position**: a logical value indicating whether the sub-triangle containing the data should be put back in the total triangle.
- **scale**: a logical value for the graph representation: the total triangle (FALSE) or the sub-triangle (TRUE).
- **min3**: if not NULL, a vector with 3 numbers between 0 and 1.
- **max3**: if not NULL, a vector with 3 numbers between 0 and 1. Let notice that `min3+max3` must equal `c(1,1,1)`.

**Author(s)**

Daniel Chessel

**Examples**

```r
if(!adegraphicsLoaded()) {
  data(euro123)
  par(mfrow = c(2, 2))
  x <- rbind.data.frame(euro123$in78, euro123$in86, euro123$in97)
  triangle.plot(x)
  triangle.class(x, as.factor(rep("G", 36)), csta = 0.5, cell = 1)
  triangle.class(x, euro123$plan$an)
  triangle.class(x, euro123$plan$pays)
  triangle.class(x, euro123$plan$an, cell = 1, axesell = TRUE)
  triangle.class(x, euro123$plan$an, cell = 0, csta = 0,
                col = c("red", "green", "blue"),
                axesell = TRUE, clab = 2, cpoi = 2)
  triangle.class(x, euro123$plan$an, cell = 2, csta = 0.5,
                axesell = TRUE, clab = 1.5)
  triangle.class(x, euro123$plan$an, cell = 0, csta = 1,
                scale = FALSE, draw.line = FALSE, show.posi = FALSE)
  par(mfrow = c(2, 2))
}
```
**triangle.plot**

**Triangular Plotting**

**Description**

Graphs for a dataframe with 3 columns of positive or null values
triangle.plot is a scatterplot
triangle.biplot is a paired scatterplots
triangle.posipoint, triangle.param, add.position.triangle are utilitaries functions.

**Usage**

triangle.plot(ta, label = as.character(1:nrow(ta)), clabel = 0,
             cpoint = 1, draw.line = TRUE, addaxes = FALSE, addmean = FALSE,
             labeltriangle = TRUE, sub = "", csub = 0, possub = "topright",
             show.position = TRUE, scale = TRUE, min3 = NULL, max3 = NULL,
             box = FALSE)

triangle.biplot(ta1, ta2, label = as.character(1:nrow(ta1)),
               draw.line = TRUE, show.position = TRUE, scale = TRUE)

**Arguments**

ta, ta1, ta2, data frame with three columns, will be transformed in **percentages** by rows
label a vector of strings of characters for the point labels
clabel if not NULL, a character size for the labels, used with par("cex")*clabel
cpoint a character size for plotting the points, used with par("cex")*cpoint. If zero, no points are drawn
draw.line a logical value indicating whether the lines into the triangle should be drawn
addaxes a logical value indicating whether the principal axes should be drawn
addmean a logical value indicating whether the mean should be plotted
labeltriangle a logical value indicating whether the variable names should be wrote
sub a string of characters to be inserted as legend
csub a character size for the legend, used with par("cex")*csub
possub a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
show.position a logical value indicating whether the used triangle should be shown in the complete one
scale a logical value indicating whether the smaller equilateral triangle containing the plot should be used
min3 If scale is FALSE, a vector of three values for the minima e.g. c(0.1,0.1,0.1) can be used
max3 If scale is FALSE a vector of three values for the maxima e.g. c(0.9,0.9,0.9) can be used
box a logical value indicating whether a box around the current plot should be drawn
Value

triangle.plot returns an invisible matrix containing the coordinates used for the plot. The graph can be supplemented in various ways.

Author(s)

Daniel Chessel

Examples

data(euro123)
tot <- rbind.data.frame(euro123$in78, euro123$in86, euro123$in97)
row.names(tot) <- paste(row.names(euro123$in78), rep(c(1, 2, 3), rep(12, 3)), sep = "")
triangle.plot(tot, label = row.names(tot), clab = 1)

par(mfrow = c(2, 2))
triangle.plot(euro123$in78, clab = 0, cpoi = 2, addmean = TRUE, show = FALSE)
triangle.plot(euro123$in86, label = row.names(euro123$in78), clab = 0.8)
triangle.biplot(euro123$in78, euro123$in86)
triangle.plot(rbind.data.frame(euro123$in78, euro123$in86), clab = 1,
               addaxes = TRUE, sub = "Principal axis", csub = 2, possub = "topright")

triangle.plot(euro123[[1]], min3 = c(0, 0.2, 0.3), max3 = c(0.5, 0.7, 0.8),
               clab = 1, label = row.names(euro123[[1]]), addax = TRUE)
triangle.plot(euro123[[2]], min3 = c(0, 0.2, 0.3), max3 = c(0.5, 0.7, 0.8),
               clab = 1, label = row.names(euro123[[1]]), addax = TRUE)
triangle.plot(euro123[[3]], min3 = c(0, 0.2, 0.3), max3 = c(0.5, 0.7, 0.8),
               clab = 1, label = row.names(euro123[[1]]), addax = TRUE)
triangle.plot(rbind.data.frame(euro123[[1]], euro123[[2]], euro123[[3]]))

par(mfrow = c(1, 1))
wtriangleplot <- cbind.data.frame(a = runif(100), b = runif(100), c = runif(100, 4, 5))
wtriangleplot <- triangle.plot(wtriangleplot)
points(wtriangleplot, col = "blue", cex = 2)
wtriangleplot <- colMeans(wtriangleplot)
points(wtriangleplot[1], wtriangleplot[2], pch = 20, cex = 3, col = "red")
rm(wtriangleplot)
Format

trichometeo is a list of 3 components.

fau is a data frame with 49 rows (trapping nights) and 17 species.

meteo is a data frame with 49 rows and 11 meteorological variables.

cla is a factor of 12 levels for the definition of the consecutive night groups

Source

Data from P. Usseglio-Polatera

References


Examples

data(trichometeo)
faulog <- log(trichometeo$fau + 1)
pca1 <- dudi.pca(trichometeo$meteo, scan = FALSE)
niche1 <- niche(pca1, faulog, scan = FALSE)

if(adegraphicsLoaded()) {
  g1 <- s.distri(niche1$ls, faulog, plab.cex = 0.6, ellipseSize = 0, starSize = 0.3, plot = FALSE)
  g2 <- s.arrow(7 * niche1$c1, plab.cex = 1, plot = FALSE)
  G <- superpose(g1, g2, plot = TRUE)
} else {
  s.label(niche1$ls, clab = 0)
  s.distri(niche1$ls, faulog, clab = 0.6, add.p = TRUE, cell = 0, csta = 0.3)
  s.arrow(7 * niche1$c1, clab = 1, add.p = TRUE)
}

ungulates

Phylogeny and quantitative traits of ungulates.

Description

This data set describes the phylogeny of 18 ungulates as reported by Pélabon et al. (1995). It also gives 4 traits corresponding to these 18 species.

Usage

data(ungulates)
uniquewt.df

**Format**

fission is a list containing the 2 following objects:
- **tre** is a character string giving the phylogenetic tree in Newick format.
- **tab** is a data frame with 18 species and 4 traits

**Details**

Variables of `ungulates$tab` are the following ones:
- afbw: is a numeric vector that describes the adult female body weight (g)
- mnw: is a numeric vector that describes the male neonatal weight (g)
- fnw: is a numeric vector that describes the female neonatal weight (g)
- ls: is a numeric vector that describes the litter size

**Source**


**Examples**

```r
data(ungulates)
ung.phy <- newick2phylog(ungulates$tre)
plot(ung.phy, clabel.l = 1.25, clabel.n = 0.75)
ung.x <- log(ungulates$tab[, 1])
ung.y <- log((ungulates$tab[, 2] + ungulates$tab[, 3]) / 2)
names(ung.x) <- names(ung.phy$leaves)
names(ung.y) <- names(ung.x)
plot(ung.x, ung.y)
abline(lm(ung.y ~ ung.x))
symbols.phylog(ung.phy, ung.x - mean(ung.x))
dotchart.phylog(ung.phy, ung.x, cle = 1.5, cno = 1.5, cdot = 1)
if (requireNamespace("adephylo", quietly = TRUE) & requireNamespace("ape", quietly = TRUE)) {
  tre <- ape::read.tree(text = ungulates$tre)
adephylo::orthogram(ung.x, tre)
  ung.z <- residuals(lm(ung.y ~ ung.x))
names(ung.z) <- names(ung.phy$leaves)
dotchart.phylog(ung.phy, ung.z, cle = 1.5, cno = 1.5, cdot = 1, ceti = 0.75)
adephylo::orthogram(ung.z, tre)
}
```

---

**uniquewt.df**  
**Elimination of Duplicated Rows in a Array**

**Description**

An utility function to eliminate the duplicated rows in a array.
Usage

uniquewt.df(x)

Arguments

x a data frame which contains duplicated rows

Value

The function returns a y which contains once each duplicated row of x.
y is an attribut 'factor' which gives the number of the row of y in which each row of x is found
y is an attribut 'length.class' which gives the number of duplicates in x with an attribut of each row of y with an attribut

Author(s)

Daniel Chessel

Examples

data(ecomor)
forsub.r <- uniquewt.df(ecomor$forsub)
attr(forsub.r, "factor")
forsub.r[1,]
ecomor$forsub[126,] #idem

dudi.pca(ecomor$forsub, scale = FALSE, scann = FALSE)$eig
# [1] 0.36845 0.24340 0.15855 0.09052 0.07970 0.04490
w1 <- attr(forsub.r, "len.class") / sum(attr(forsub.r,"len.class"))
dudi.pca(forsub.r, row.w = w1, scale = FALSE, scann = FALSE)$eig
# [1] 0.36845 0.24340 0.15855 0.09052 0.07970 0.04490

variance.phylog

The phylogenetic ANOVA

Description

This function performs the variance analysis of a trait on eigenvectors associated to a phylogenetic tree.

Usage

variance.phylog(phylog, z, bynames = TRUE, na.action = c("fail", "mean"))
Arguments

- `phylog`: an object of class `phylog`
- `z`: a numeric vector of the values corresponding to the variable
- `bynames`: if TRUE checks if `z` labels are the same as `phylog` leaves label, possibly in a different order. If FALSE the check is not made and `z` labels must be in the same order than `phylog` leaves label
- `na.action`: if ‘fail’ stops the execution of the current expression when `z` contains any missing value. If ‘mean’ replaces any missing values by mean(`z`)

Details

- `phylog$Amat` defines a set of orthonormal vectors associated the each nodes of the phylogenetic tree.
- `phylog$Adim` defines the dimension of the subspace `A` defined by the first `phylog$Adim` vectors of `phylog$Amat` that corresponds to phylogenetic inertia.
- `variance.phylog` performs the linear regression of `z` on `A`.

Value

- Returns a list containing
  - `lm`: an object of class `lm` that corresponds to the linear regression of `z` on `A`.
  - `anova`: an object of class `anova` that corresponds to the anova of the precedent model.
  - `smry`: an object of class `anova` that is a summary of the precedent object.

Author(s)

- Sébastien Ollier <sebastien.ollier@u-psud.fr>
- Daniel Chessel

References


See Also

- `phylog`, `lm`

Examples

```r
data(njplot)
njplot.phy <- newick2phylog(njplot$tre)
variance.phylog(njplot.phy,njplot$tauxcg)
par(mfrow = c(1,2))
table.phylog(njplot.phy$Ascores, njplot.phy, clabel.row = 0,
```
```r
clabel.col = 0.1, clabel.nod = 0.6, csize = 1)
dotchart.phylog(njplot.phy, njplot$tauxcg, clabel.nodes = 0.6)
if (requireNamespace("adephylo", quietly = TRUE) & requireNamespace("ape", quietly = TRUE)) {
  tre <- ape::read.tree(text = njplot$tre)
adephylo::orthogram(njplot$tauxcg, tre = tre)
}
```

---

### varipart

**Partition of the variation of a response multivariate table by 2 explanatory tables**

#### Description

The function partitions the variation of a response table (usually community data) with respect to two explanatory tables. The function performs the variation partitioning based on redundancy analysis (RDA, if `dudiY` is obtained by `dudi.pca`) or canonical correspondence analysis (CCA, if `dudiY` is obtained by `dudi.coa`) and computes unadjusted and adjusted R-squared. The significance of R-squared are evaluated by a randomization procedure where the rows of the explanatory tables are permuted.

#### Usage

```r
varipart(Y, X, W = NULL, nrepet = 999, type = c("simulated", "parametric"),
scale = FALSE, ...)
```  
```
## S3 method for class 'varipart'
print(x, ...)
```

#### Arguments

**Y**  
a vector, matrix or data frame or an object of class `dudi`. If not a `dudi` object, the data are treated by a principal component analysis (`dudi.pca`).

**X, W**  
dataframes or matrices of explanatory (co)variables (numeric and/or factor variables). By default, no covariables are considered (`W` is `NULL`) and this case corresponds to simple canonical ordination.

**nrepet**  
an integer indicating the number of permutations.

**type**  
a character specifying the algorithm which should be used to adjust R-squared (either "simulated" or "parametric").

**scale**  
If `Y` is not a `dudi`, a logical indicating if variables should be scaled

**...**  
进一步的参数传递给 `as.krandtest` 或 `as.randtest` (如果无 covariables 被考虑) 对于函数 `varipart`。

**x**  
an object of class `varipart`
Details

Two types of algorithm are provided to adjust R-squared. The "simulated" procedure estimates the unadjusted R-squared expected under the null hypothesis H0 and uses it to adjust the observed R-squared as follows: \( R_{2, \text{adj}} = 1 - \frac{(1 - R^2)}{(1 - E(R^2|H0))} \) with \( R_{2, \text{adj}} \) the adjusted R-squared and \( R^2 \) the unadjusted R-squared. The "parametric" procedure performs the Ezequiel's adjustment on the unadjusted R-squared as: \( R_{2, \text{adj}} = 1 - \frac{(1 - R^2)}{(1 - p / (n - 1))} \) where n is the number of sites, and p the number of predictors.

Value

It returns an object of class varipart. It is a list with:

- test: the significance test of fractions \([ab], [bc], \text{and} [abc]\) based on randomization procedure. An object of class krandtest
- R2: unadjusted estimations of fractions \([a], [b], [c], \text{and} [d]\)
- R2adj: adjusted estimations of fractions \([a], [b], [c], \text{and} [d]\)
- call: the matched call

Author(s)

Stephane Dray <stephane.dray@univ-lyon1.fr> and Sylvie Clappe <sylvie.clappe@univ-lyon1.fr>

References


See Also

pcaiv

Examples

data(mafragh)

# PCA on response table Y
Y <- mafragh$flo
dudiY <- dudi.pca(Y, scannf = FALSE, scale = FALSE)

# Variation partitioning based on RDA
# without covariables
vprda <- varipart(dudiY, mafragh$env)

# Variation partitioning based on RDA
# with covariables and parametric estimation
vprda <- varipart(dudiY, mafragh$env, mafragh$xy, type = "parametric")
Description

This data set contains abundance values (Braun-Blanquet scale) of 80 plant species for 337 sites. Data have been collected by Sonia Said and Francois Debias.

Usage

data(vegtf)

Format

vegtf is a list with the following components:

  veg  a data.frame with the abundance values of 80 species (columns) in 337 sites (rows)
  xy   a data.frame with the spatial coordinates of the sites
  area a data.frame (area) which define the boundaries of the study site
  sp.names a vector containing the species latin names
  nb   a neighborhood object (class nb defined in package spdep)
  Spatial an object of the class SpatialPolygons of sp, containing the map

Source


Examples

if(requireNamespace("spdep", quietly = TRUE)) {
  data(vegtf)
  coa <- dudi.coa(vegtf$veg, scannf = FALSE)
  ms.coa <- multispati(coa, listw = spdep::nb2listw(vegtf$nb), nfposi = 2,
                       nfnega = 0, scannf = FALSE)
  summary(ms.coa)
  plot(ms.coa)

  if(adegraphicsLoaded()) {
    g1 <- s.value(vegtf$xy, coa$li[, 1], Sp = vegtf$Spatial, pSp.col = "white", plot = FALSE)
    g2 <- s.value(vegtf$xy, ms.coa$li[, 1], Sp = vegtf$Spatial, pSp.col = "white", plot = FALSE)
    g3 <- s.label(coa$c1, plot = FALSE)
    g4 <- s.label(ms.coa$c1, plot = FALSE)
    G <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
  } else {
    
  }
Example for Centring in PCA

Description
The data come from the INSEE (National Institute of Statistics and Economical Studies). It is an array of widower percentages in relation with the age and the socioprofessional category.

Usage
data(veuvage)

Format
veuvage is a list of 2 components.

   tab is a data frame with 37 rows (widowers) 6 columns (socio-professional categories)
   age is a vector of the ages of the 37 widowers.

Details
The columns contain the socioprofessional categories:
1- Farmers, 2- Craftsmen, 3- Executives and higher intellectual professions,
4- Intermediate Professions, 5- Others white-collar workers and 6- Manual workers.

Source
unknown

Examples
data(veuvage)
par(mfrow = c(3,2))
for (j in 1:6) plot(veuvage$age, veuvage$tab[,j],
xlab = "age", ylab = "pourcentage de veufs",
type = "b", main = names(veuvage$tab)[j])
Within-Class Analysis

Description
Perform a particular case of an Orthogonal Principal Component Analysis with respect to Instrumental Variables (orthopcaiv), in which there is only a single factor as covariable.

Usage
## S3 method for class 'dudi'
wca(x, fac, scannf = TRUE, nf = 2, ...)

Arguments
- `x`: a duality diagram, object of class `dudi` from one of the functions `dudi.coa`, `dudi.pca`,...
- `fac`: a factor partitioning the rows of `dudi$tab` in classes
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: if `scannf` FALSE, an integer indicating the number of kept axes
- `...`: further arguments passed to or from other methods

Value
Returns a list of the sub-class `within` in the class `dudi`
- `tab`: a data frame containing the transformed data (subtraction of the class mean)
- `call`: the matching call
- `nf`: number of kept axes
- `rank`: the rank of the analysis
- `ratio`: percentage of within-class inertia
- `eig`: a numeric vector containing the eigenvalues
- `lw`: a numeric vector of row weights
- `cw`: a numeric vector of column weights
- `tabw`: a numeric vector of class weights
- `fac`: the factor defining the classes
- `li`: data frame row coordinates
- `l1`: data frame row normed scores
- `co`: data frame column coordinates
- `c1`: data frame column normed scores
- `ls`: data frame supplementary row coordinates
- `as`: data frame inertia axis onto within axis
Note

To avoid conflict names with the `base:::within` function, the function `within` is now deprecated and removed. It is replaced by the method `wca.dudi` of the new generic `wca` function.

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References


Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
wit1 <- wca(pca1, meaudret$design$site, scan = FALSE, nf = 2)

if(adegraphicsLoaded()) {
  g1 <- s.traject(pca1$li, meaudret$design$site, psub.text = "Principal Component Analysis",
                  plines.lty = 1:nlevels(meaudret$design$site), psub.cex = 1.5, plot = FALSE)
  g2 <- s.traject(wit1$li, meaudret$design$site,
                  psub.text = "Within site Principal Component Analysis",
                  plines.lty = 1:nlevels(meaudret$design$site), psub.cex = 1.5, plot = FALSE)
  g3 <- s.corcircle (wit1$as, plot = FALSE)
  G <- ADEgS(list(g1, g2, g3), layout = c(2, 2))
} else {
  par(mfrow = c(2, 2))
  s.traject(pca1$li, meaudret$design$site, sub = "Principal Component Analysis", csub = 1.5)
  s.traject(wit1$li, meaudret$design$site, sub = "Within site Principal Component Analysis",
            csub = 1.5)
  s.corcircle (wit1$as)
  par(mfrow = c(1,1))
}
plot(wit1)

wca.coinertia

Within-class coinertia analysis

Description

Performs a within-class analysis after a coinertia analysis
Usage

## S3 method for class 'coinertia'
wca(x, fac, scannf = TRUE, nf = 2, ...)

Arguments

- **x**: a coinertia analysis (object of class `coinertia`) obtained by the function `coinertia`
- **fac**: a factor partitioning the rows in classes
- **scannf**: a logical value indicating whether the eigenvalues barplot should be displayed
- **nf**: if scannf FALSE, an integer indicating the number of kept axes
- **...**: further arguments passed to or from other methods

Details

This analysis is equivalent to do a within-class analysis on each initial dudi, and a coinertia analysis on the two within analyses. This function returns additional outputs for the interpretation.

Value

An object of the class `witcoi`. Outputs are described by the `print` function.

Note

To avoid conflict names with the `base:::within` function, the function `within` is now deprecated and removed. To be consistent, the `withincoinertia` function is also deprecated and is replaced by the method `wca.coinertia` of the generic `wca` function.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr> and Jean Thioulouse <jean.thioulouse@univ-lyon1.fr>

References


See Also

coinertia, wca

Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
pca2 <- dudi.pca(meaudret$spe, scal = FALSE, scan = FALSE, nf = 4)
wit1 <- wca(pca1, meaudret$design$site, scan = FALSE, nf = 2)
wit2 <- wca(pca2, meaudret$design$site, scan = FALSE, nf = 2)
coiw <- coinertia(wit1, wit2, scannf = FALSE)
wca.rlq

Within-Class RLQ analysis

Description

Performs a particular RLQ analysis where a partition of sites (rows of R) is taken into account. The within-class RLQ analysis search for linear combinations of traits and environmental variables of maximal covariance.

Usage

```r
## S3 method for class 'rlq'
wca(x, fac, scannf = TRUE, nf = 2, ...)
## S3 method for class 'witrlq'
plot(x, xax = 1, yax = 2, ...)
## S3 method for class 'witrlq'
print(x, ...)
```

Arguments

- `x`: an object of class rlq (created by the rlq function) for the wca.rlq function. An object of class witrlq for the print and plot functions.
- `fac`: a factor partitioning the rows of R.
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed.
- `nf`: if scannf FALSE, an integer indicating the number of kept axes.
- `xax`: the column number for the x-axis.
- `yax`: the column number for the y-axis.
- `...`: further arguments passed to or from other methods.

Value

The wca.rlq function returns an object of class 'betrlq' (sub-class of 'dudi'). See the outputs of the print function for more details.

Author(s)

Stéphane Dray <stephane.dray@univ-lyon1.fr>
References


See Also

*rlq*, *wca*, *wca.rlq*

Examples

data(piosphere)
afcL <- dudi.coa(log(piosphere$veg + 1), scannf = FALSE)
acpR <- dudi.pca(piosphere$env, scannf = FALSE, row.w = afcL$lw)
acpQ <- dudi.hillsmith(piosphere$traits, scannf = FALSE, row.w = afcL$cw)
rlq1 <- rlq(acpR, afcL, acpQ, scannf = FALSE)
wrlq1 <- wca(rlq1, fac = piosphere$habitat, scannf = FALSE)
wrlq1
plot(wrlq1)

---

**westafrica**  
*Freshwater fish zoogeography in west Africa*

Description

This data set contains informations about faunal similarities between river basins in West Africa.

Usage

data(westafrica)

Format

*westafrica* is a list containing the following objects:

- **tab** : a data frame with absence/presence of 268 species (rows) at 33 embouchures (columns)
- **spe.names** : a vector of string of characters with the name of species
- **spe.binames** : a data frame with the genus and species (columns) of the 256 species (rows)
- **riv.names** : a vector of string of characters with the name of rivers
- **atlantic** : a data frame with the coordinates of a polygon that represents the limits of atlantic (see example)
- **riv.xy** : a data frame with the coordinates of embouchures
- **lines** : a data frame with the coordinates of lines to complete the representation (see example)
- **cadre** : a data frame with the coordinates of points used to make the representation (see example)
within

Within-Class Analysis

Description

Outputs and graphical representations of the results of a within-class analysis.

Usage

```r
## S3 method for class 'within'
plot(x, xax = 1, yax = 2, ...)  
## S3 method for class 'within'
print(x, ...)  
## S3 method for class 'witcoi'
plot(x, xax = 1, yax = 2, ...)  
## S3 method for class 'witcoi'
print(x, ...)  
## S3 method for class 'within'
summary(object, ...)  
```
Arguments

x, object  an object of class within or witcoi
xax      the column index for the x-axis
yax      the column index for the y-axis
...    further arguments passed to or from other methods

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>
Stéphane Dray <stephane.dray@univ-lyon1.fr>

References


See Also

wca.dudi, wca.coinertia

Examples

data(meaudret)
pca1 <- dudi.pca(meaudret$env, scan = FALSE, nf = 4)
wit1 <- wca(pca1, meaudret$design$site, scan = FALSE, nf = 2)

if(adegraphicsLoaded()) {
g1 <- s.traject(pca1$li, meaudret$design$site, psub.text = "Principal Component Analysis", plines.lty = 1:length(levels(meaudret$design$site)), plot = FALSE)
g2 <- s.traject(wit1$li, meaudret$design$site, psub.text = "Within site Principal Component Analysis", plines.lty = 1:length(levels(meaudret$design$site)), plot = FALSE)
g3 <- s.corcircle (wit1$as, plot = FALSE)
G <- ADEgS(list(g1, g2, g3), layout = c(2, 2))
} else {
par(mfrow = c(2, 2))
s.traject(pca1$li, meaudret$design$site, sub = "Principal Component Analysis", csub = 1.5)
s.traject(wit1$li, meaudret$design$site, sub = "Within site Principal Component Analysis", csub = 1.5)
s.corcircle (wit1$as)
par(mfrow = c(1, 1))
}

plot(wit1)
withinpca  

Normed within principal component analysis

Description

Performs a normed within Principal Component Analysis.

Usage

```r
withinpca(df, fac, scaling = c("partial", "total"),
          scannf = TRUE, nf = 2)
```

Arguments

- `df`: a data frame with quantitative variables
- `fac`: a factor partitioning the rows of `df` in classes
- `scaling`: a string of characters as a scaling option:
  - if "partial", the sub-table corresponding to each class is centred and normed.
  - If "total", the sub-table corresponding to each class is centred and the total table is then normed.
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: if scannf FALSE, an integer indicating the number of kept axes

Details

This function implements the 'Bouroche' standardization. In a first step, the original variables are standardized (centred and normed). Then, a second transformation is applied according to the value of the `scaling` argument. For "partial", variables are standardized in each sub-table (corresponding to each level of the factor). Hence, variables have null mean and unit variance in each sub-table. For "total", variables are centred in each sub-table and then normed globally. Hence, variables have a null mean in each sub-table and a global variance equal to one.

Value

returns a list of the sub-class `within` of class `dudi`. See `wca`

Author(s)

Daniel Chessel
Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr>

References

Examples

```r
data(meaudret)
wit1 <- withinpca(meaudret$env, meaudret$design$season, scannf = FALSE, scaling = "partial")
ktal <- ktab.within(wit1, colnames = rep(c("S1", "S2", "S3", "S4", "S5"), 4))
unclass(ktal)

# See pta
plot(wit1)
```

### witwit.coa

**Internal Correspondence Analysis**

#### Description

`witwit.coa` performs an Internal Correspondence Analysis. `witwitsepan` gives the computation and the barplot of the eigenvalues for each separated analysis in an Internal Correspondence Analysis.

#### Usage

```r
witwit.coa(dudi, row.blocks, col.blocks, scannf = TRUE, nf = 2)
## S3 method for class 'witwit'
summary(object, ...)
witwitsepan(ww, mfrow = NULL, csub = 2, plot = TRUE)
```

#### Arguments

- `dudi`: an object of class `coa`
- `row.blocks`: a numeric vector indicating the row numbers for each block of rows
- `col.blocks`: a numeric vector indicating the column numbers for each block of columns
- `scannf`: a logical value indicating whether the eigenvalues bar plot should be displayed
- `nf`: if `scannf` FALSE, an integer indicating the number of kept axes
- `object`: an object of class `witwit`
- `...`: further arguments passed to or from other methods
- `ww`: an object of class `witwit`
- `mfrow`: a vector of the form "c(nr,nc)", otherwise computed by a special own function 'n2mfrow'
- `csub`: a character size for the sub-titles, used with `par("cex")*csub`
- `plot`: if FALSE, numeric results are returned
Value

returns a list of class witwit, coa and dudi (see as.dudi) containing

- `rbvar` a data frame with the within variances of the rows of the factorial coordinates
- `lbw` a data frame with the marginal weighting of the row classes
- `cvar` a data frame with the within variances of the columns of the factorial coordinates
- `cbw` a data frame with the marginal weighting of the column classes

Author(s)

Daniel Chessel Anne-Béatrice Dufour <anne-beatrice.dufour@univ-lyon1.fr> Correction by Campo Elías PARDO <cepardot@cable.net.co>

References


Examples

data(ardeche)
coa1 <- dudi.coa(ardeche$tab, scann = FALSE, nf = 4)
ww <- witwit.coa(coa1, ardeche$row.blocks, ardeche$col.blocks, scann = FALSE)
summary(ww)

if(adegraphicsLoaded()) {
  g1 <- s.class(ww$co, ardeche$sta.fac, plab.cex = 1.5, ellipseSi = 0, paxes.draw = FALSE, plot = FALSE)
  g2 <- s.label(ww$co, plab.cex = 0.75, plot = FALSE)
  G <- superpose(g1, g2, plot = TRUE)
} else {
  s.class(ww$co, ardeche$sta.fac, clab = 1.5, cell = 0, axesell = FALSE)
  s.label(ww$co, add.p = TRUE, clab = 0.75)
}

witwitsepan(ww, c(4, 6))

woangers

Plant assemblages in woodlands of the conurbation of Angers (France)

Description

This data set gives the presence of plant species in relevés of woodlands in the conurbation of Angers; and their biological traits.
Usage
data(woangers)

Format

woangers is a list of 2 components.

1. flo: is a data frame that contains the presence/absence of species in each sample site. In the
codes for the sample sites (first column of the data frame), the first three letters provide the
code of the woodland and the numbers represent the 5 quadrats sampled in each site. Codes
for the woodlands are based on either their local name when they have one or on the name of
the nearest locality.

2. traits: is a data frame that contains the values of the 13 functional traits considered in the
paper. One trait can be encoded by several columns. The codes are as follows:
   • Column 1: Species names;
   • Column 2: li, nominal variable that indicates the presence (y) or absence (n) of ligneous
     structures;
   • Column 3: pr, nominal variable that indicates the presence (y) or absence (n) of prickly
     structures;
   • Column 4: fo, circular variable that indicates the month when the flowering period starts
     (from 1 January to 9 September);
   • Column 5: he, ordinal variable that indicates the maximum height of the leaf canopy;
   • Column 6: ae, ordinal variable that indicates the degree of aerial vegetative multiplication;
   • Column 7: un, ordinal variable that indicates the degree of underground vegetative
     multiplication;
   • Column 8: lp, nominal variable that represents the leaf position by 3 levels (ros = rosette,
     semiros = semi-rosette and leafy = leafy stem);
   • Column 9: le, nominal variable that represents the mode of leaf persistence by 5 levels
     (seasae = seasonal aestival, seashib = seasonal hibernal, seasver = seasonal vernal,
     everalw = always evergreen, everparti = partially evergreen);
   • Columns 10, 11 and 12: fuzzy variable that describes the modes of pollination with 3
     levels (auto = autopollination, insects = pollination by insects, wind = pollination by
     wind); this fuzzy variable is expressed as proportions, i.e. for each row, the sum of the
     three columns equals 1;
   • Columns 13, 14 and 15: fuzzy variable that describes the life cycle with 3 levels (annual,
     monocarpic and polycarpic); this fuzzy variable is expressed as proportions, i.e. for each
     row, the sum of the three column equals 1;
   • Columns 16 to 20: fuzzy variable that describes the modes of dispersion with 5 levels
     (elaio = dispersion by ants, endozoo = injection by animals, epizoo = external transport
     by animals, wind = transport by wind, unsp = unspecialized transport); this fuzzy variable
     is expressed as proportions, i.e. for each row, the sum of the three columns equals 1;
   • Column 21: lo, quantitative variable that provides the seed bank longevity index;
   • Column 22: lf, quantitative variable that provides the length of the flowering period.
Examples

# Loading the data
data(woangers)

# Preparing of the traits
traits <- woangers$traits
# Nominal variables 'li', 'pr', 'lp' and 'le'
# (see table 1 in the main text for the codes of the variables)
tabN <- traits[, c(1:2, 7, 8)]
# Circular variable 'fo'
tabC <- traits[3]
tabCp <- prep.circular(tabC, 1, 12)
# The levels of the variable lie between 1 (January) and 12 (December).
# Ordinal variables 'he', 'ae' and 'un'
tabO <- traits[, 4:6]
# Fuzzy variables 'mp', 'pe' and 'di'
tabF <- traits[, 9:19]
tabFp <- prep.fuzzy(tabF, c(3, 3, 5), labels = c("mp", "pe", "di"))
# 'mp' has 3 levels, 'pe' has 3 levels and 'di' has 5 levels.
# Quantitative variables 'lo' and 'lf'
tabQ <- traits[, 20:21]

# Combining the traits
ktab1 <- ktab.list.df(list(tabN, tabCp, tabO, tabFp, tabQ))
## Not run:
# Calculating the distances for all traits combined
distrait <- dist.ktab(ktab1, c("N", "C", "O", "F", "Q"))
is.euclid(distrait)

# Calculating the contribution of each trait in the combined distances
contrib <- kdist.cor(ktab1, type = c("N", "C", "O", "F", "Q"))
contrib
dotchart(sort(contrib$glocor), labels = rownames(contrib$glocor)[order(contrib$glocor[, 1])])

## End(Not run)

worksurv

French Worker Survey (1970)

Description

The worksurv data frame gives 319 response items and 4 questions providing from a French Worker Survey.
Usage

data(worksurv)

Format

This data frame contains the following columns:

1. pro: Professional elections. In professional elections in your firm, would you rather vote for a list supported by?
   - CGT
   - CFDT
   - FO
   - CFTC
   - Auton Autonomous
   - Abst
   - Nonaffi Not affiliated
   - NR No response

2. una: Union affiliation. At the present time, are you affiliated to a Union, and in the affirmative, which one?
   - CGT
   - CFDT
   - FO
   - CFTC
   - Auton Autonomous
   - CGC
   - Notaffi Not affiliated
   - NR No response

3. pre: Presidential election. On the last presidential election (1969), can you tell me the candidate for whom you have voted?
   - Duclos
   - Deferre
   - Krivine
   - Rocard
   - Poher
   - Ducatel
   - Pompidou
   - NRAds No response, abstention

4. pol: political sympathy. Which political party do you feel closest to, as a rule?
   - Communist (PCF)
   - Socialist (SFIO+PSU+FGDS)
   - Left (Party of workers,...)
   - Center MRP+RAD.
The data frame `worksurv` has the attribute `counts` giving the number of responses for each item.

**Source**


**References**


**Examples**

```r
data(worksurv)
acm1 <- dudi.acm(worksurv, row.w = attr(worksurv, "counts"), scan = FALSE)

if(adegraphicsLoaded()) {
  s.class(acm1$li, worksurv)
} else {
  par(mfrow = c(2, 2))
  apply(worksurv, 2, function(x) s.class(acm1$li, factor(x), attr(worksurv, 'counts')))
  par(mfrow = c(1, 1))
}
```

**yanomama**

**Distance Matrices**

**Description**

This data set gives 3 matrices about geographical, genetic and anthropometric distances.

**Usage**

```r
data(yanomama)
```

**Format**

yanomama is a list of 3 components:

- `geo` is a matrix of 19-19 geographical distances
- `gen` is a matrix of 19-19 SFA (genetic) distances
- `ant` is a matrix of 19-19 anthropometric distances
Source

References

Examples
data(yanomama)
gen <- quasieuclid(as.dist(yanomama$gen)) # depends of mva
ant <- quasieuclid(as.dist(yanomama$ant)) # depends of mva
par(mfrow = c(2,2))
plot(gen, ant)
t1 <- mantel.randtest(gen, ant, 99);
plot(t1, main = "gen-ant-mantel") ; print(t1)
t1 <- procuste.rtest(pcoscaled(gen), pcoscaled(ant), 99)
plot(t1, main = "gen-ant-procuste") ; print(t1)
t1 <- RV.rtest(pcoscaled(gen), pcoscaled(ant), 99)
plot(t1, main = "gen-ant-RV") ; print(t1)

Road distances in New-Zealand

Description
This data set gives the road distances between 13 towns in New-Zealand.

Usage
data(zealand)

Format
zealand is a list with the following components:

- **road** a data frame with 13 rows (New Zealand towns) and 13 columns (New Zealand towns) containing the road distances between these towns
- **xy** a data frame containing the coordinates of the 13 towns
- **neig** an object of class `neig`, a neighbour graph to visualize the map shape
- **nb** a neighborhood object (class `nb` defined in package `spdep`)
Examples

data(zealand)
d0 <- as.dist(as.matrix(zealand$road))
d1 <- cailliez(d0)
d2 <- lingoes(d0)

if(adegraphicsLoaded()) {
  G1 <- s.label(zealand$xy, lab = as.character(1:13), nb = zealand$nb)
  g1 <- s.label(cmdscale(dist(zealand$xy)), lab = as.character(1:13), nb = zealand$nb,
                psub.text = "Distance canonique", plot = FALSE)
  g2 <- s.label(cmdscale(d0), lab = as.character(1:13), nb = zealand$nb,
                psub.text = "Distance routiere", plot = FALSE)
  g3 <- s.label(cmdscale(d1), lab = as.character(1:13), nb = zealand$nb,
                psub.text = "Distance routiere / Cailliez", plot = FALSE)
  g4 <- s.label(cmdscale(d2), lab = as.character(1:13), nb = zealand$nb,
                psub.text = "Distance routiere / Lingoes", plot = FALSE)
  G2 <- ADEgS(list(g1, g2, g3, g4), layout = c(2, 2))
} else {
  s.label(zealand$xy, lab = as.character(1:13), neig = zealand$neig)
  s.label(cmdscale(dist(zealand$xy)), lab = as.character(1:13),
           neig = zealand$neig, sub = "Distance canonique", csub = 2)
  s.label(cmdscale(d0), lab = as.character(1:13), neig = zealand$neig,
           sub = "Distance routiere", csub = 2)
  s.label(cmdscale(d1), lab = as.character(1:13), neig = zealand$neig,
           sub = "Distance routiere / Cailliez", csub = 2)
  s.label(cmdscale(d2), lab = as.character(1:13), neig = zealand$neig,
           sub = "Distance routiere / Lingoes", csub = 2)
}
Index

* ~bca
  loocv.discrimin, 172
* ~dplot
  loocv.discrimin, 172
* ~multivariate
  loocv.discrimin, 172
* array
  cailliez, 50
dist.binary, 79
dist.dudi, 80
dist.neig, 84
dist.prop, 85
dist.quant, 86
dudi.pco, 110
is.euclid, 140
lingoes, 168
mantel.randtest, 181
mantel.rtest, 182
pcoscaled, 241
quasieuclid, 260
* chron
  arrival, 21
* datasets
  abouheif.eg, 8
acacia, 9
aminoacyl, 13
apis108, 15
aravo, 17
ardeche, 18
arrival, 21
atlas, 23
atya, 25
avijons, 26
avimedi, 28
aviurba, 29
bacteria, 30
banque, 31
baran95, 34
bf88, 42
bordeaux, 44
bsetal97, 44
buech, 46
butterfly, 47
capitaes, 51
carni19, 52
carni70, 53
carniherbi49, 54
casitas, 55
chatcat, 56
chats, 57
chazeb, 58
chevaine, 59
chickenk, 60
clementines, 61
cnc2003, 62
coleo, 65
corvus, 69
deug, 74
doubs, 93
dunedata, 112
ecg, 112
ecomor, 113
elec88, 115
escopage, 117
euro123, 118
fission, 119
friday87, 125
fruits, 126
ggtortoises, 129
granulo, 131
hdpg, 133
housetasks, 134
humDNAm, 135
ichtyo, 136
irishdata, 138
julliot, 141
jv73, 143
kcponds, 144
INDEX

lascaux, 166
lizards, 169
macaca, 176
macon, 177
macroloire, 177
mafragh, 179
maples, 183
mariages, 184
meau, 193
meaudret, 195
microsatt, 198
mjrochet, 199
mollusc, 202
monde84, 203
morphosport, 204
newick.eg, 216
njplot, 224
olympic, 225
oribatid, 226
ours, 231
palm, 233
pap, 234
pcw, 242
perthio2, 243
piosphere, 247
presid2002, 250
procella, 252
rankrock, 272
rhizobium, 274
rhone, 275
rpjdl, 279
santacatalina, 310
sarcelles, 311
seconde, 334
skulls, 336
steppe, 341
syndicats, 350
t3012, 351
tarentaise, 358
taxo.eg, 359
tintooool, 362
tithonia, 363	
tortues, 364
toxicity, 365
trichometeo, 369
ungulates, 370
vegtf, 376
veuvage, 377
westafrica, 382
woangers, 388
worksurv, 390
yanomama, 392
zealand, 393

*dplot
loocv.between, 170
loocv.dudi, 174

*hplot
add.scatter, 10
area.plot, 19
dotchart.phylog, 90
dotcircle, 92
kplot, 150
kplot.foucart, 151
kplot.mcoa, 152
kplot.mfa, 153
kplot.pta, 154
kplot.sepan, 155
kplot.statis, 157
plot.phylog, 248
s.arrow, 284
s.chull, 286
s.class, 287
s.corcircle, 289
s.distri, 291
s.hist, 293
s.image, 294
s.kde2d, 296
s.label, 297
s.logo, 299
s.match, 301
s.match.class, 302
s.multinom, 304
s.trajec, 306
s.value, 308
scatter, 313
scatter.acm, 314
scatter.coa, 315
scatter.dudi, 316
scatter.fca, 317
scatterutil, 318
sco.boxplot, 320
sco.class, 321
sco.distri, 322
sco.gauss, 324
sco.label, 325
sco.match, 326
INDEX

sco.quant, 328
score, 329
score.acm, 330
score.coa, 331
score.mix, 332
score.pca, 333
symbols.phylog, 349
table.cont, 352
table.dist, 353
table.paint, 354
table.phylog, 355
table.value, 356
triangle.class, 366
triangle.plot, 368

* htest
  randboot, 261
  randxval, 271

* manip
  ade4-package, 8
  as.taxo, 22
  newick2phylog, 217
  phylog, 244
  PI2newick, 246

* methods
  krandtest, 158
  randtest, 263
  rtest, 280

* models
  variance.phylog, 372

* multivariate
  add.scatter, 10
  ade4-package, 8
  amova, 14
  apqe, 16
  bca, 36
  bca.coinertia, 38
  bca.rlg, 39
  between, 40
  bwca.d pca, 48
  coinertia, 63
  combine.4thcorner, 66
  costatis, 70
  costatis.randtest, 71
disc, 75
discrimin, 76
discrimin.coa, 78
dist.binary, 79
dist.dudi, 80
dist.ktab, 81
dist.neig, 84
dist.prop, 85
dist.quant, 86
dive, 88
divcmax, 89
dpcoa, 94
dudi, 96
dudi.acm, 98
dudi.coa, 100
dudi.dec, 101
dudi.fca, 102
dudi.hillsmith, 104
dudi.mix, 106
dudi.nsc, 107
dudi.pca, 108
dudi.pco, 110
foucart, 120
fourthcorner, 122
inertia.dudi, 137
kdist, 146
kdist2ktab, 148
kdisteuclid, 149
kplot, 150
kplot.foucart, 151
kplot.mcoa, 152
kplot.mfa, 153
kplot.pta, 154
kplot.sepan, 155
kplot.stats, 157
ktab, 159
ktab.data.frame, 161
ktab.list.df, 162
ktab.list.dudi, 163
ktab.match2ktabs, 164
ktab.within, 165
lingoes, 168
loocv.between, 170
loocv.dudi, 174
mbpcaiv, 185
mbpils, 187
mcoa, 189
mdpcoa, 190
mfa, 196
multiblock, 206
multispati, 207
multispati.randtest, 210
multispati.rtest, 212
niche, 219
nipals, 221
originality, 227
pcaiv, 235
pcaivortho, 239
procuste, 253
procuste.randtest, 256
procuste.rtest, 257
pta, 258
randboot.multiblock, 262
randtest.amova, 264
randtest.between, 265
randtest.coinertia, 266
randtest.discrimin, 267
randtest.pcaiv, 270
reconst, 272
rlq, 276
rtest.between, 281
rtest.discrimin, 282
RV.rtest, 283
RVdist.randtest, 284
s.arrow, 284
s.chull, 286
s.class, 287
s.corcircle, 289
s.distri, 291
s.hist, 293
s.kde2d, 296
s.label, 297
s.logo, 299
s.match, 301
s.match.class, 302
s.multinom, 304
s.traject, 306
s.value, 308
scatter, 313
scatter.acm, 314
scatter.coa, 315
scatter.dudi, 316
scatter.fca, 317
scatterutil, 318
sco.boxplot, 320
sco.class, 321
sco.distri, 322
sco.gauss, 324
sco.label, 325
sco.match, 326
sco.quant, 328
score, 329
score.acm, 330
score.coa, 331
score.mix, 332
score.pca, 333
sepan, 335
statico, 337
statico.krandtest, 338
statis, 339
supcol, 342
supdist, 343
suprow, 345
suprow.pta, 347
testdim, 360
testdim.multiblock, 361
wca, 378
wca.coinertia, 379
wca.rlq, 381
within, 384
withinpca, 386
witwit.coa, 387
* nonparametric
corkdist, 68
costatis.randtest, 71
mantel.randtest, 181
mantel.rtest, 182
multispati.randtest, 210
multispati.rtest, 212
procuste.randtest, 256
procuste.rtest, 257
randtest.amova, 264
randtest.between, 265
randtest.coinertia, 266
randtest.discrimin, 267
randtest.pcaiv, 270
rtest.between, 281
rtest.discrimin, 282
RV.rtest, 283
RVdist.randtest, 284
* spatial
gearymoran, 128
gridrowcol, 132
mld, 200
multispati, 207
multispati.randtest, 210
multispati.rtest, 212
orthobasis, 228
rlq, 276
| **ts** | *gearymoran, 128* |
| | *mld, 200* |
| | *orthobasis, 228* |
| **utilities** | *bicenter.wt, 43* |
| | *kdisteuclid, 149* |
| | *mstree, 205* |
| | *neig, 213* |
| | *scalewt, 312* |
| | *uniquewt.df, 371* |
| [.dudi | *as.randboot (randboot), 261* |
| | as.randtest (randtest), 263* |
| | as.randxval (randxval), 271* |
| | as.taxo, 22, 217* |
| | atlas, 23* |
| | atya, 25* |
| | avijons, 26* |
| | avimedi, 28* |
| | aviurba, 29* |
| | *bacteria, 30* |
| | banque, 31* |
| | baran95, 34* |
| | bca, 36, 39, 40* |
| | bca.coinertia, 38, 41* |
| | bca.dpcoa (bwca.dpcoa), 48* |
| | bca.dudi, 41* |
| | bca.rlq, 39* |
| | between, 40* |
| | bf88, 42* |
| | bicenter.wt, 43* |
| | biplot.dudi (scatter), 313* |
| | bordeaux, 44* |
| | boxplot.acm (dudi.acm), 98* |
| | bssetal197, 44* |
| | buech, 46* |
| | butterfly, 47* |
| | bwca.dpcoa, 48* |
| | *c.kdist (kdist), 146* |
| | c.ktab (ktab), 159* |
| | cailliez, 50* |
| | capitales, 51* |
| | carni19, 52* |
| | carni70, 53* |
| | carniherbi49, 54* |
| | casitas, 55* |
| | chatcat, 56* |
| | chats, 57* |
| | chazeb, 58* |
| | chevaine, 59* |
| | chickenk, 60* |
| | circ.plot, 92* |
| | clementines, 61* |
| | cnc2003, 62* |
| | coinertia, 38, 39, 63, 70, 278, 339, 380* |
| | col.names (ktab), 159* |
| | col.names<- (ktab), 159* |
| | coleo, 65* |
INDEX

combine.4thcorner, 66, 125
combine.randtest.r1q
  (combine.4thcorner), 66
corkdist, 68
corvus, 69
costatis, 70
costatis.randtest, 71
covfacwt (scalewt), 312
covwt (scalewt), 312
dagnelie.test, 72
daisy, 83
Depreciated functions, 74
deg, 74
disc, 75
discrimin, 76, 78
discrimin.coa, 78
dist.binary, 79
dist.dudi, 80
dist.ktab, 81
dist.neig, 84
dist.prop, 85
dist.quant, 86
dist.taxon (as.taxon), 22
divc, 88
divcmax, 89
dotchart.phylog, 90, 349
dotcircle, 92
doubs, 93
dpcoa, 49, 94, 190, 193, 269
dudi, 36, 96, 99, 100, 102, 103, 105, 106, 108, 109, 111, 208, 211, 212, 378
dudi.acm, 98
dudi.coa, 100
dudi.dec, 101
dudi.fca, 102
dudi.fpca (dudi.fca), 102
dudi.hillsmith, 104
dudi.mix, 106
dudi.nsc, 107
dudi.pca, 108, 223, 361
dudi.pco, 110, 344
dunedata, 112
ecg, 112
ecomor, 113
elec88, 115
enum.phylog (plot.phylog), 248
escopage, 117
euro123, 118
fission, 119
foucart, 120
fourthcorner, 67, 122, 278
fourthcorner2 (fourthcorner), 122
friday87, 125
fruits, 126
geary.test, 129
geary.moran, 128
ggtortoises, 129
granulo, 131
ggridrowcol, 132, 201, 230
haar2level (mld), 200
hclust2phylog (newick2phylog), 217
hdpg, 133
housetasks, 134
humDNAm, 135
ichtyo, 136
inertia (inertia.dudi), 137
inertia.dudi, 137
irishdata, 138
is.dudi (dudi), 96
is.euclid, 140
is.ktab (ktab), 159
is.orthobasis (orthobasis), 228
julliot, 141
tv73, 143
kcponds, 144
kdist, 146
kdist.cor (dist.ktab), 81
kdist2ktab, 148
kdisteuclid, 149
kplot, 150
kplot.foucart, 151
kplot.mcoa, 152
kplot.mfa, 153
kplot.pta, 154
kplot.sepan, 155
kplot.stats, 157
kplotsepan.coa (kplot.sepan), 155
kplotX.mdpcoa (mdpcoa), 190
krandtest, 158, 339
krandxval (randxval), 271
ktab, 159, 162, 163, 165, 166, 337
ktab.data.frame, 160, 161
ktab.list.df, 160, 162
ktab.list.dudi, 160, 163
ktab.match2ktabs, 160, 164, 337
ktab.within, 160, 165

lascaux, 166
ldist.ktab (dist.ktab), 81
lingoes, 168
lizards, 169
lm, 373
loocv (loocv.dudi), 174
loocv.between, 170, 173, 175
loocv.discrimin, 171, 172, 175
loocv.dudi, 171, 173, 174

macaca, 176
macon, 177
macroloire, 177
mafgragh, 179
mantel.randtest, 181, 264
mantel.rtest, 182, 280
mantelkdist (corkdist), 68
maples, 183
mariages, 184
mat2listw, 208, 211, 212
mbpcaiv, 185, 206, 262, 263, 361, 362
mbpls, 186, 187, 188, 206, 262, 263, 361, 362
mcoa, 189
mdpcoa, 190
meanfacwt (scalewt), 312
meau, 193, 195
meaudret, 194, 195
mfa, 196
microsatt, 198
mjrochet, 199
mlg, 132, 200, 230
mollusc, 202
monde84, 203
moran.test, 129
morphosport, 204
mra, 201
mstree, 205
multiblock, 206
multispati, 207
multispati.randtest, 210
multispati.rtest, 212

nb2neig (neig), 213
neig, 213
neig2mat (neig), 213
neig2nb (neig), 213
newick.eg, 216
newick2phylog, 217, 245
nich, 219
nipals, 221
njplot, 224
olympic, 225
oribatid, 226
originality, 227
orthobasis, 132, 201, 228
orthogram, 132, 201, 230
ours, 231

p.adjust.4thcorner (combine.4thcorner), 66
p.adjust.methods, 66, 67, 123, 125
palm, 233
pap, 234
pcaiv, 235, 375
pcaivortho, 239
pcscaled, 241
pcw, 242
perthi02, 243
phylog, 217, 230, 244, 249, 373
PI2newick, 246
piosphere, 247
plot.4thcorner (fourthcorner), 122
plot.betcoi (between), 40
plot.betrlq (bca.rlq), 39
plot.between (between), 40
plot.coinertia (coinertia), 63
plot.corkdist (corkdist), 68
plot.discrimin (discrimin), 76
plot.dpcoa (dpcoa), 94
plot.foucart (foucart), 120
plot.krtest (krandtest), 158
plot.mcoa (mcoa), 189
plot.mfa (mfa), 196
plot.multispati (multispati), 207
plot.niche (nich), 219
plot.orthobasis (orthobasis), 228
plot.pcaiv (pcaiv), 235
plot.phylog, 217, 245, 248
plot.procuste (procuste), 253
plot.pta (pta), 258
plot.randtest (randtest), 263
plot.rlq(rlq), 276
plot.sepan(sepan), 335
plot.stats(statis), 339
plot.witcoi(within), 384
plot.within(within), 384
plot.witrlq(wca.rlq), 381
poly2area(area.plot), 19
predict.dudi(suprow), 345
prep.binary(dist.ktab), 81
prep.circular(dist.ktab), 81
prep.fuzzy(var(dudi.fca)), 102
prep.mdpcoa(mdpcosa), 190
presid2002, 250
print.4thcorner(fourthcorner), 122
print.amova(amova), 14
print.apqe(apqe), 16
print.betcoi(between), 40
print.betdpcoa(bwca.dpcoa), 48
print.betrlq(betrlq), 39
print.between(between), 40
print.coinertia(coinertia), 63
print.corkdist(corkdist), 68
print.discrimin(discrimin), 76
print.dpcoa(dpcoa), 94
print.dudi(dudi), 96
print.foucart(foucart), 120
print.inertia(inertia.dudi), 137
print.kdist(kdist), 146
print.krandboot(randboot), 261
print.krandtest(krandtest), 158
print.krandxval(randxval), 271
print.ktab(ktab), 159
print.mcoa(mcoa), 189
print.mfa(mfa), 196
print.multiblock(multiblock), 206
print.multispati(multispati), 207
print.neig(neig), 213
print.niche(niche), 219
print.nipals(nipals), 221
print.orthobasis(orthobasis), 228
print.pcaiv(pcaiv), 235
print.phylog(phylog), 244
print.procuste(procuste), 253
print.pta(pta), 258
print.randboot(randboot), 261
print.randtest(randtest), 263
print.randxval(randxval), 271
print.rlq(rlq), 276
print.sepan(sepan), 335
print.statis(statis), 339
print.varipart(varipart), 374
print.witcoi(within), 384
print.witdpcoa(bwca.dpcoa), 48
print.within(within), 384
print.witrlq(wca.rlq), 381
procolla, 252
procella, 252
procuste, 253
procuste.randtest, 256, 264
procuste.randtest, 257, 280
pta, 70, 164, 258, 337
quasieuclid, 260
radial.phylog(plot.phylog), 248
randboot, 261
randboot.multiblock, 186, 188, 262, 262, 362
randtest, 159, 263, 280, 339
randtest.amova, 15, 264
randtest.between, 265
randtest.betwit(bwca.dpcoa), 48
randtest.coinertia, 266
randtest.discrimin, 267
randtest.dpcoa, 268
randtest.pcaiv, 270
randtest.pcaivortho(randtest.pcaiv), 270
randtest.procuste(procuste), 253
randtest.rlx(rlq), 276
randxval, 271
rankrock, 272
reciprocal.coa(score.coa), 331
reconst, 272
redo.dudi(dudi), 96
rhizobium, 274
rhone, 275
rlq, 40, 67, 125, 276, 382
row.names.ktab(ktab), 159
row.names<-.ktab(ktab), 159
rpjdl, 279
rtest, 264, 280
rtest.between, 281
rtest.discrimin, 282
rtest.niche(niche), 219
RV.rtest, 280, 283, 361
RVDist.randtest, 284
INDEX

RVkdist (corkdist), 68
s.arrow, 284, 314, 319
s.chull, 99, 286, 314, 319
s.class, 99, 287, 304, 314, 319
s.circircle, 289, 314, 319
s.distri, 291, 314, 319
s.hist, 293
s.image, 294
s.kde2d, 296
s.label, 297, 314, 319
s.logo, 299
s.match, 175, 301, 304, 314, 319
s.match.class, 302
s.multinom, 304
s.traject, 306, 314, 319
s.value, 308, 314, 319
santacatalina, 310
sarcelles, 311
scalefacwt (scalewt), 312
scalewt, 312
scatter, 11, 313
scatter.acm, 314
scatter.coa, 315
scatter.dudi, 316
scatter.fca, 317
scatter.nipals (nipals), 221
scatter.pco (dudi.pco), 110
scatterutil, 318
scatterutil.logo (s.logo), 299
sco.boxplot, 320, 329
sco.class, 321
sco.distri, 322, 329
sco.gauss, 324
sco.label, 325
sco.match, 326
sco.quant, 328, 329
score, 329
score.acm, 330
score.coa, 331
score.mix, 332
score.pca, 333
scores.neig (neig), 213
scoreutil.base (score), 329
scoreplot.dudi (scatter), 313
seconde, 334
sepan, 335
skulls, 336
statico, 337
statico.krandtest, 338
stats, 339
steppe, 341
summary.4thcorner (fourthcorner), 122
summary.between (between), 40
summary.betwit (bwca.dpca), 48
summary.coinertia (coinertia), 63
summary.corkdist (corkdist), 68
summary.dist (is.euclid), 140
summary.dpcoa (dpca), 94
summary.dudi (dudi), 96
summary.inertia (inertia.dudi), 137
summary.mcoa (mcoa), 189
summary.mfa (mfa), 196
summary.multiblock (multiblock), 206
summary.multispati (multispati), 207
summary.neig (neig), 213
summary.orthobasis (orthobasis), 228
summary.pcaiv (pcaiv), 235
summary.pcaivortho (pcaivortho), 239
summary.rlq (rlq), 276
summary.sepan (sepan), 335
summary.within (within), 384
summary.witwit (witwit.coa), 387
supcol, 342
supdist, 343
suprow, 175, 344, 345
suprow.pta, 347
symbols.phylog, 91, 349, 356
syndicats, 350

t.dudi (dudi), 96
t.ktab (ktab), 159
t3012, 351
tab.names (ktab), 159
tab.names<- (ktab), 159
table.cont, 352
table.dist, 353
table.paint, 354
table.phylog, 91, 349, 355
table.prepare (table.value), 356
table.value, 356
tarentaise, 358
taxo.eg, 359
taxo2phylog, 23
taxo2phylog (newick2phylog), 217
testdim, 360
testdim.multiblock, 186, 188, 263, 271, 361
tintoodiel, 362
tithonia, 363
tortues, 364
toxicity, 365
triangle.biplot (triangle.plot), 368
triangle.class, 366
triangle.param (triangle.plot), 368
triangle.plot, 368
triangle.posipoint (triangle.plot), 368
trichometeo, 369

ungulates, 370
uniquewt.df, 371

varfacwt (scalewt), 312
variance.phylog, 372
varipart, 374
varwt (scalewt), 312
vegtf, 376
veuvage, 377

wavelet.filter, 229
wca, 378, 380, 382, 386
wca.coinertia, 379, 385
wca.dpcoa (bwca.dpcoa), 48
wca.dudi, 385
wca.r1q, 40, 381, 382
westafrica, 382
within, 384
withinpca, 386
witwit.coa, 387
witwitsepan (witwit.coa), 387
woangers, 83, 388
worksurv, 390

yanomama, 392

zealand, 393