# Package ‘hierfstat’

## July 20, 2020

**Version**  0.5-7  
**Date**  2020-07-20  
**Title** Estimation and Tests of Hierarchical F-Statistics  
**Author** Jerome Goudet [aut, cre],  
Thibaut Jombart [aut],  
Zhian N. Kamvar [ctb],  
Eric Archer [ctb],  
Olivier Hardy [ctb]  
**Maintainer** Jerome Goudet <jerome.goudet@unil.ch>  
**Imports** ade4, adegenet, gaston, gtools, methods  
**Suggests** ape, pegas, knitr, rmarkdown, testthat  
**Depends** R (>= 2.10)  
**License** GPL (>= 2)  
**URL** https://www.r-project.org, https://github.com/jgx65/hierfstat  
**BugReports** https://github.com/jgx65/hierfstat/issues  
**VignetteBuilder** knitr  
**RoxygenNote** 6.1.1  
**NeedsCompilation** no  
**Repository** CRAN  
**Date/Publication** 2020-07-20 18:20:11 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>3</td>
</tr>
<tr>
<td>allele.count</td>
<td>4</td>
</tr>
<tr>
<td>allelic.richness</td>
<td>5</td>
</tr>
<tr>
<td>basic.stats</td>
<td>6</td>
</tr>
<tr>
<td>beta.dosage</td>
<td>8</td>
</tr>
<tr>
<td>betas</td>
<td>9</td>
</tr>
<tr>
<td>biall2dos</td>
<td>11</td>
</tr>
<tr>
<td>boot.ppfis</td>
<td>11</td>
</tr>
<tr>
<td>boot.ppfst</td>
<td>12</td>
</tr>
<tr>
<td>boot.vc</td>
<td>13</td>
</tr>
<tr>
<td>cont.isl</td>
<td>14</td>
</tr>
<tr>
<td>cont.isl99</td>
<td>15</td>
</tr>
<tr>
<td>crocrussula</td>
<td>16</td>
</tr>
<tr>
<td>diploid</td>
<td>16</td>
</tr>
<tr>
<td>exhier</td>
<td>17</td>
</tr>
<tr>
<td>fs.dosage</td>
<td>18</td>
</tr>
<tr>
<td>fstat2dos</td>
<td>19</td>
</tr>
<tr>
<td>g.stats</td>
<td>20</td>
</tr>
<tr>
<td>g.stats.glob</td>
<td>21</td>
</tr>
<tr>
<td>genet.dist</td>
<td>23</td>
</tr>
<tr>
<td>genind2hierfstat</td>
<td>24</td>
</tr>
<tr>
<td>genot2al</td>
<td>25</td>
</tr>
<tr>
<td>getal</td>
<td>26</td>
</tr>
<tr>
<td>getal.b</td>
<td>27</td>
</tr>
<tr>
<td>grm2kinship</td>
<td>27</td>
</tr>
<tr>
<td>gtrunchier</td>
<td>28</td>
</tr>
<tr>
<td>hierfstat</td>
<td>29</td>
</tr>
<tr>
<td>ind.count</td>
<td>30</td>
</tr>
<tr>
<td>indpca</td>
<td>30</td>
</tr>
<tr>
<td>kinship2dist</td>
<td>31</td>
</tr>
<tr>
<td>kinship2grm</td>
<td>32</td>
</tr>
<tr>
<td>kinshipShift</td>
<td>33</td>
</tr>
<tr>
<td>mat2vec</td>
<td>33</td>
</tr>
<tr>
<td>ms2bed</td>
<td>34</td>
</tr>
<tr>
<td>ms2dos</td>
<td>34</td>
</tr>
<tr>
<td>nb.alleles</td>
<td>35</td>
</tr>
<tr>
<td>pairwise.neifst</td>
<td>36</td>
</tr>
<tr>
<td>pairwise.WCfst</td>
<td>37</td>
</tr>
<tr>
<td>pcoa</td>
<td>38</td>
</tr>
<tr>
<td>pop.freq</td>
<td>38</td>
</tr>
<tr>
<td>pp.fst</td>
<td>39</td>
</tr>
<tr>
<td>pp.sigma.loc</td>
<td>40</td>
</tr>
<tr>
<td>print.pp.fst</td>
<td>40</td>
</tr>
<tr>
<td>qn2.read.fstat</td>
<td>41</td>
</tr>
<tr>
<td>read.fstat</td>
<td>42</td>
</tr>
<tr>
<td>read.fstat.data</td>
<td>43</td>
</tr>
</tbody>
</table>
**AIc**

Calculates corrected Assignment Index

**Description**

Calculates corrected Assignment Index as described in Goudet et al. (2002)

**Usage**

`AIc(dat)`

**Arguments**

`dat` a data frame with nlocs+1 columns,

**Value**

`aic` The corrected assignment index of each individual

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>
References


---

allele.count

**Allele counts**

**Description**

Counts the number of copies of the different alleles at each locus and population

**Usage**

`allele.count(data,diploid=TRUE)`

**Arguments**

- **data**: A data frame containing the population of origin in the first column and the genotypes in the following ones
- **diploid**: Whether the data are from diploid individuals

**Value**

A list of tables, --each with np (number of populations) columns and nl (number of loci) rows-- of the count of each allele

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**Examples**

```
data(gtrunchier)
allele.count(gtrunchier[,,-2])
```
Allelic richness Estimates allelic richness

Description
Estimates allelic richness, the rarefied allelic counts, per locus and population

Usage
`allelic.richness(data, min.n=NULL, diploid=TRUE)`

Arguments
- **data**: A data frame, with as many rows as individuals. The first column contains the population to which the individual belongs, the following to the different loci
- **min.n**: The number of alleles down to which the number of alleles should be rarefied. The default is the minimum number of individuals genotyped (times 2 for diploids)
- **diploid**: a boolean specifying whether individuals are diploid (default) or haploid

Value
- **min.all**: The number of alleles used for rarefaction
- **Ar**: A table with as many rows as loci and columns as populations containing the rarefied allele counts

Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

References

Examples
``` R
data(gtrunchier)
allelic.richness(gtrunchier[, -1])
```
Description

Estimates individual counts, allelic frequencies, observed heterozygosities and genetic diversities per locus and population. Also Estimates mean observed heterozygosities, mean gene diversities within population Hs, Gene diversities overall Ht and corrected Htp, and Dst, Dstp. Finally, estimates Fst and Fstp as well as Fis following Nei (1987) per locus and overall loci.

Usage

```
basic.stats(data,diploid=TRUE,digits=4)
## S3 method for class 'basic.stats'
print(x,...)
```

Arguments

data a data frame where the first column contains the population to which the different individuals belong, and the following columns contain the genotype of the individuals -one locus per column-
diploid Whether individuals are diploids (default) or haploids
digits how many digits to print out in the output (default is 4)
x an object of class basic.stats
... further arguments to pass to print.bas.stats

Value

- `n.ind.samp` A table –with np (number of populations) columns and nl (number of loci) rows– of genotype counts
- `pop.freq` A list containing allele frequencies. Each element of the list is one locus. For each locus, Populations are in columns and alleles in rows
- `Ho` A table –with np (number of populations) columns and nl (number of loci) rows– of observed heterozygosities
- `Hs` A table –with np (number of populations) columns and nl (number of loci) rows– of observed gene diversities
- `Fis` A table –with np (number of populations) columns and nl (number of loci) rows– of observed Fis
- `perloc` A table –with as many rows as loci– containing basic statistics Ho, Hs, Ht, Dst, Ht', Dst', Fst, Fst', Fis, Dest
- `overall` Basic statistics averaged over loci
Note

For the perloc and overall tables (see value section), the following statistics, defined in eq.7.38–7.43 pp.164–5 of Nei (1987) are estimated:

The observed heterozygosity

$$H_o = 1 - \sum_k \sum_i P_{kii}/np,$$

where $P_{kii}$ represents the proportion of homozygote $i$ in sample $k$ and $np$ the number of samples.

The within population gene diversity (sometimes misleadingly called expected heterozygosity):

$$H_s = \tilde{n}/(n - 1)[1 - \sum_i \bar{p}_i^2 - H_o/2\tilde{n}],$$

where $\tilde{n} = np/\sum_k 1/n_k$ and $\bar{p}_i^2 = \sum_k p_{ki}^2/np$

The overall gene diversity

$$H_t = 1 - \sum_i \bar{p}_i^2 + H_s/(\tilde{n}np) - H_o/(2\tilde{n}np),$$

where $\bar{p}_i = \sum_k p_{ki}/np$.

The amount of gene diversity among samples $Dst = H_t - H_s$

$$Dst' = np/(np - 1)Dst$$

$$Ht' = H_s + Dst'$$

$Fst = Dst/Ht$ (This is not the same as Nei’s $Gst$, Nei’s $Gst$ is an estimator of $Fst$ based on allele frequencies only)

$$Fst' = Dst'/Ht'$$

$Fis = 1 - H_o/H_s$

Last, $Dest = np/(np - 1)(Ht' - H_s)/(1 - H_s)$ a measure of population differentiation as defined by Jost (2008) is also given

Here, the $p_{ki}$ are unweighted by sample size. These statistics are estimated for each locus and an overall loci estimates is also given, as the unweighted average of the per locus estimates. In this way, monomorphic loci are accounted for (with estimated value of 0) in the overall estimates.

Note that the equations used here all rely on genotypic rather than allelic number and are corrected for heterozygosity.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

References


See Also

`ind.count`, `pop.freq`.

Examples

```r
data(gtrunchier)
basic.stats(gtrunchier[,-1])
```

---

**beta.dosage**

Estimates pairwise kinships and individual inbreeding coefficients from dosage data

Description

Estimates pairwise kinships (coancestries) and individual inbreeding coefficient using Weir and Goudet (2017) beta estimator.

Usage

```r
beta.dosage(dos,inb=TRUE,Mb=FALSE)
```

Arguments

- `dos`: A matrix of 0, 1 and 2s with loci (SNPs) in columns and individuals in rows. missing values are allowed
- `inb`: whether individual inbreeding coefficient should be estimated (rather than self-coancestries)
- `Mb`: whether to output the mean matching

Details

This function is written for dosage data, i.e., how many doses of an allele (0, 1 or 2) an individual carries. It should be use for bi-allelic markers only (e.g. SNPs), although you might "force" a k multiallelic locus to k biallelic loci.

Matching proportion can be obtained by the following equation: $M = \beta \ast (1 - Mb) + Mb$

By default (inb=TRUE) the inbreeding coefficient is returned on the main diagonal. With inb=FALSE, self coancestries are reported.

Twice the betas with self-coancestries on the diagonal gives the Genomic Relationship Matrix (GRM)

Following a suggestion from Olivier Hardy, missing data are removed from the estimation procedure, rather than imputed (this is taken care off automatically)
Value

if $M_b=$FALSE, a matrix of pairwise kinships and inbreeding coefficients (if inb=TRUE) or self-coancestries (inb=FALSE); if $M_b=$FALSE, a list with elements inb (whether inbreeding coefficients rather than kinships should be returned on the main diagonal), $M_b$ (the average off-diagonal matching) and $\beta$s the kinships or inbreeding coefficients.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

References


Examples

## Not run:
dos<-matrix(sample(0:2,size=10000,replace=TRUE),ncol=100)
beta.dosage(dos,inb=TRUE)
## End(Not run)

betas Estimates $\beta$s per population and a bootstrap confidence interval

Description

Estimate populations (Population specific FST) or individual coancestries and a bootstrap confidence interval, assuming random mating

Usage

betas(dat,nboot=0,lim=c(0.025,0.975),diploid=TRUE,betaijT=FALSE)

Arguments

dat data frame with genetic data and pop identifier
nboot number of bootstrap samples.
lim width of the bootstrap confidence interval
diploid whether the data comes from a diploid organism
betaijT whether to estimate individual coancestries
Details

If betaijT=TRUE, and the first column contains a unique identifier for each individual, the function returns the matrix of individual coancestries/kinships. Individual inbreeding coefficients can be obtained by multiplying by 2 the diagonal and subtracting 1.

Value

Hi Within population gene diversities (complement to 1 of matching probabilities)
Hb Between populations gene diversities
betaiovl Average $\beta_i$ over loci (Population specific FSTs)
betaW Average of the betaiovl (overall population FST)
ci The bootstrap confidence interval of population specific FSTs (only if more than 100 bootstraps requested and if more than 10 loci are present)

if betaijT=TRUE, return the matrix of pairwise coancestries only.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

References

Weir and Goudet  A unified characterization of population structure and relatedness.

Examples

```r
## Not run:
#3 different population sizes lead to 3 different betais
dat<-sim.genot(size=40,N=c(50,200,1000),nbloc=50,nbal=10)
betas(dat,nboot=100)

#individual coancestries from the smallest population are large
ind.coan<-betas(cbind(1:120,dat[,1]),betaij=T)
graphics::image(1:120,1:120,ind.coan,xlab="Inds",ylab="Inds")

#extracting individual inbreeding coefficients
dat<-sim.genot(size=20,nbloc=100,nbal=20,mig=0.01,f=c(0,0.3,0.7))
ind.coan<-betas(cbind(1:60,dat[,1]),betaijT=TRUE)
ind.inb<-(diag(ind.coan)*2-1)
graphics::boxplot(ind.inb~dat[,1], main="Individual inbreeding coefficients")
graphics::points(1:3,c(0,0.3,0.7),pch=16,col="red",cex=2)

## End(Not run)
```
**biall2dos**  
*Converts bi-allelic SNPs from hierfstat format to dosage format*

**Description**

Converts bi-allelic SNPs hierfstat format to dosage format, the number of alternate allele copies at a locus for an individual, i.e. 11 -> 0; 12 or 21 -> 1 and 22 -> 2

**Usage**

```
biall2dos(dat, diploid=TRUE)
```

**Arguments**

- `dat`: a hierfstat data frame without the first column (the population identifier), individuals in rows, columns with individual genotypes encoded as 11, 12, 21 and 22
- `diploid`: whether the data set is from a diploid organism

**Value**

a matrix containing allelic dosages

**Examples**

```
## Not run:
biall2dos(sim.genot(nbal=2,nbloc=10)[,-1])  # a 10 column matrix

## End(Not run)
```

---

**boot.ppfis**  
*Performs bootstrapping over loci of population’s Fis*

**Description**

Performs bootstrapping over loci of population’s Fis

**Usage**

```
boot.ppfis(dat=dat,nboot=100,quant=c(0.025, 0.975), diploid=TRUE, dig=4,...)
```
Arguments

dat a genetic data frame
nboot number of bootstraps
quant quantiles
diploid whether diploid data
dig digits to print
... further arguments to pass to the function

Value

call function call
fis.ci Bootstrap ci of Fis per population

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

dat<-sim.genot(nbpop=4,nbloc=20,nbal=10,f=c(0,0.2,0.4,0.6))
boot.ppfis(dat)

Description

Performs bootstrapping over loci of pairwise Fst

Usage

boot.ppfst(dat=dat,nboot=100,quant=c(0.025,0.975),diploid=TRUE,...)

Arguments

dat a genetic data frame
nboot number of bootstraps
quant the quantiles for bootstrapped ci
diploid whether data are from diploid organisms
... further arguments to pass to the function
**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>call to the function</td>
</tr>
<tr>
<td>ll</td>
<td>lower limit ci</td>
</tr>
<tr>
<td>ul</td>
<td>upper limit ci</td>
</tr>
<tr>
<td>vc.per.loc</td>
<td>for each pair of population, the variance components per locus</td>
</tr>
</tbody>
</table>

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**Examples**

```r
data(gtrunchier)
x<-boot.ppfst(gtrunchier[,-2])
x$ll
x$ul
```

**Description**

Provides a bootstrap confidence interval (over loci) for sums of the different variance components (equivalent to gene diversity estimates at the different levels), and the derived F-statistics, as suggested by Weir and Cockerham (1984). Will not run with less than 5 loci. Raymond and Rousset (199X) points out shortcomings of this method.

**Usage**

```r
boot.vc(levels=levels,loci=loci,diploid=TRUE,nboot=1000,quant=c(0.025,0.5,0.975))
```

**Arguments**

- `levels`: a data frame containing the different levels (factors) from the outermost (e.g. region) to the innermost before the individual
- `loci`: a data frame containing the different loci
- `diploid`: Specify whether the data are coming from diploid or haploid organisms (diploid is the default)
- `nboot`: Specify the number of bootstrap to carry out. Default is 1000
- `quant`: Specify which quantile to produce. Default is c(0.025,0.5,0.975) giving the percentile 95% CI and the median
Value

boot a data frame with the bootstrapped variance components. Could be used for obtaining bootstrap ci of statistics not listed here.
res a data frame with the bootstrap derived statistics. H stands for gene diversity, F for F-statistics
ci Confidence interval for each statistic.

References


See Also

varcomp.glob.

Examples

#load data set
data(gtrunchier)
boot.vc(gtrunchier[,c(1:2)],gtrunchier[,-c(1:2)],nboot=100)

cont.isl A genetic dataset from a diploid organism in a continent-island model

Description

A simple diploid dataset, with allele encoded as one digit number. Up to 4 alleles per locus

Usage

data(cont.isl)

Format

A data frame with 150 rows and 6 columns:

Pop Population identifier, from 1 to 3
loc.1 genotype at loc.1
loc.2 genotype at loc.2
loc.3 genotype at loc.3
loc.4 genotype at loc.4
loc.5 genotype at loc.5
...
cont.isl99

Source

generated with function sim.genot()

Examples

data(cont.isl)
allele.count(cont.isl)

---

cont.isl99  A genetic dataset from a diploid organism in a continent-island model

Description

A simple diploid dataset, with alleles encoded as two digits numbers. Up to 99 alleles per locus

Usage

data(cont.isl99)

Format

A data frame with 150 rows and 6 columns:

- **Pop**  Population identifier, from 1 to 3
- **loc.1**  genotype at loc.1
- **loc.2**  genotype at loc.2
- **loc.3**  genotype at loc.3
- **loc.4**  genotype at loc.4
- **loc.5**  genotype at loc.5
  ...

Source

generated with function sim.genot(nbal=99)

Examples

data(cont.isl99)
allele.count(cont.isl99)
crocrussula  
*Genotypes and sex of 140 shrews Crocidura russula*

**Description**
A dataset containing microsatellite genotypes, population and sex of 140 Crocidura russula individuals

**Usage**
data(crocrussula)

**References**

**Examples**
data(crocrussula)
aic<-A1c(crocrussula$genot)
boxplot(aic~crocrussula$sex)
sexbias.test(crocrussula$genot,crocrussula$sex)

diploid  
*A genetic dataset from a diploid organism*

**Description**
A simple diploid dataset, with allele encoded as one digit number

**Usage**
data(diploid)

**Format**
A data frame with 44 rows and 6 columns:

- **Pop** Population identifier, from 1 to 6
- **loc-1** genotype at loc-1 (only allele 4 present)
- **loc-2** genotype at loc-1 (alleles 3 and 4)
**loc-3** genotype at loc-1 (alleles 2, 3 and 4)
**loc-4** genotype at loc-1 (alleles 1, 2, 3 and 4)
**loc-5** genotype at loc-1 (only allele 4)

**Source**
Given in Weir, B.S. Genetic Data Analysis. Sinauer

**Examples**
```
data(diploid)
basic.stats(diploid)
```

---

**Description**
Example data set with 4 levels, one diploid and one haploid locus

**Usage**
```
data(exhier)
```

**Value**
```
lev1    outermost level
lev2    level 2
lev3    Level 3
lev4    Level 4
diplo   Diploid locus
haplo   Haploid locus
```

**Examples**
```
data(exhier)
varcomp(exhier[,1:5])
varcomp(exhier[,c(1:4,6)],diploid=FALSE)
```
fs.dosage  

Estimates F-statistics from dosage data

Description

Reports individual inbreeding coefficients, Population specific Fsts and Fiss from dosage data

Usage

fs.dosage(dos, pop, matching = FALSE)

## S3 method for class 'fs.dosage'
plot(x, ...)

## S3 method for class 'fs.dosage'
print(x, digits = 4, ...)

fst.dosage(dos, pop, matching = FALSE)

fis.dosage(dos, pop, matching = FALSE)

Arguments

dos  
either a matrix with snps columns and individuals in rows containing allelic
dosage (number [0,1 or 2] of alternate alleles); or a square matrix with as many
rows and columns as the number of individuals and containing the proportion of
matching alleles

pop  
a vector containing the identifier of the population to which the individual in the
corresponding row belongs

matching  
logical:TRUE if dos is a square matrix of allelic matching; FALSE otherwise

x  
a fs.dosage object

...  
further arguments to pass

digits  
number of digits to print

Value

Fi list of individual inbreeding coefficients, estimated with the reference being the population to
which the individual belongs.

FsM matrix containing population specific FSTs on the diagonal. The off diagonal elements con-
tains the average of the kinships for pairs of individuals, one from each population, relative to the
mean kinship for pairs of individuals between populations.

Fs The first row contains population specific and overall Fis, the second row population specific and
overall Fst.
Methods (by generic)

- plot: Plot function for fs.dosage class
- print: Print function for fs.dosage class

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>


Examples

```r
## Not run:
dos<-matrix(sample(0:2,size=10000,replace=TRUE),ncol=100)
fs.dosage(dos,pop=rep(1:5,each=20))
plot(fs.dosage(dos,pop=rep(1:5,each=20)))
## End(Not run)
```

fstat2dos

Converts a hierfstat genetic data frame to dosage data

Description

Converts a hierfstat genetic data frame to dosage. For each allele at each locus, allelic dosage (number of copies of the allele) is reported. The column name is the allele identifier.

Usage

`fstat2dos(dat,diploid=TRUE)`

Arguments

- `dat`: data frame with genetic data without the first column (population identifier)
- `diploid`: whether the data set is from a diploid organism

Value

A matrix with \( \sum l \ n_l^a \) columns (where \( n_l^a \) is the number of alleles at locus \( l \)), as many rows as individuals, and containing the number of copies (dosage) of the corresponding allele.
Examples

```r
## Not run:
dat<-sim.genot(nbal=5,nbloc=10)
dos<-fstat2dos(dat[,,-1])
dim(dos)
wcdat)
fstat.dosage(dos,pop=dat[,1])

## End(Not run)
```

---

### g.stats

Calculates likelihood-ratio $G$-statistic on contingency table

#### Description

Calculates the likelihood ratio $G$-statistic on a contingency table of alleles at one locus $X$ sampling unit. The sampling unit could be any hierarchical level.

#### Usage

```r
g.stats(data,diploid=TRUE)
```

#### Arguments

- **data**: a two-column data frame. The first column contains the sampling unit, the second the genotypes.
- **diploid**: Whether the data are from diploid (default) organisms.

#### Value

- **obs**: Observed contingency table.
- **exp**: Expected number of allelic observations.
- **X.squared**: The chi-squared statistics, $\sum \frac{(O-E)^2}{E}$.
- **g.stats**: The likelihood ratio statistics, $2 \sum (O \log \frac{O}{E})$.

#### Author(s)

Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland

<jerome.goudet@unil.ch>
g.stats.glob

References

See Also
g.stats.glob.

Examples
data(gtrunchier)
attach(gtrunchier)
g.stats(data.frame(Patch,L21.V))

---

**g.stats.glob**

*Likelihood ratio G-statistic over loci*

**Description**
Calculates the likelihood ratio G-statistic on a contingency table of alleles at one locus X sampling unit, and sums this statistic over the loci provided. The sampling unit could be any hierarchical level (patch, locality, region,...). By default, diploid data are assumed

**Usage**
g.stats.glob(data,diploid=TRUE)

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>a data frame made of n1+1 column, n1 being the number of loci. The first column contains the sampling unit, the others the multi-locus genotype. Only complete multi-locus genotypes are kept for calculation</td>
</tr>
<tr>
<td>diploid</td>
<td>Whether the data are from diploid (default) organisms</td>
</tr>
</tbody>
</table>

**Value**

g.stats.l  Per locus likelihood ratio statistic

**Author(s)**
Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland
<jerome.goudet@unil.ch>
References


See Also

`g.stats`, `samp.within`, `samp.between`.

Examples

data(gtrunchier)
attach(gtrunchier)
nperm <- 99
nobs <- length(Patch)
gglobs.o <- vector(length=(nperm+1))
gglobs.p <- vector(length=(nperm+1))
gglobs.l <- vector(length=(nperm+1))

for (i in 1:nperm) {#careful, might take a while
  gglobs.o[i] <- g.stats.glob(data.frame(Patch, gtrunchier[,c(1,2)]))$g.stats
  gglobs.p[i] <- g.stats.glob(data.frame(Patch, gtrunchier[,c(1,2)]))$g.stats
  gglobs.l[i] <- g.stats.glob(data.frame(Locality, gtrunchier[,c(1,2)]))$g.stats
}

#p-value of first test (among patches)
p.globs.o <- sum(gglobs.o>=gglobs.o[nperm+1])/(nperm+1)

#p-value of second test (among patches within localities)
p.globs.p <- sum(gglobs.p>=gglobs.p[nperm+1])/(nperm+1)

#p-value of third test (among localities)
p.globs.l <- sum(gglobs.l>=gglobs.l[nperm+1])/(nperm+1)

#Are alleles associated at random among patches
p.globs.o

#Are alleles associated at random among patches within localities?
#Tests differentiation among patches within localities
p.globs.p

#Are alleles associated at random among localities, keeping patches as one unit?
#Tests differentiation among localities
genet.dist

Description

Estimates one of several genetic distances among all pairs of populations.

Usage

```r
genet.dist(dat, diploid=TRUE, method="Dch")
```

Arguments

- `dat`: A data frame containing population of origin as the first column and multi-locus genotypes in following columns.
- `diploid`: Whether the data is from a diploid (default) or haploid organism.

Details

The method argument specify which genetic distance to use, among eight, all briefly described in Takezaki and Nei (1996)

- “Dch” By default, Cavalli-Sforza and Edwards Chord distance (eqn 6 in the reference) is returned. This distance is used as default since Takezaki & Nei (1996) found that it was the best to retrieve the relation among samples.
- “Da” This is Nei’s et al genetic distance (eqn 7), performing nearly as well as “Dch”
- “Ds” Nei’s standard genetic distance (eqn 1). Increases linearly with divergence time but has larger variance
- “Fst” Latter’s and also approximately Reynolds et al Genetic distance (eqn 3)
- “Dm” Nei’s minimum distance (eqn 2)
- “Dr” Rogers’s distance (eqn 4)
- “Cp” Prevosti et al’s distance (eqn 5)
- “X2” Sanghvi’s distance (eqn 8)
- “Nei87” see pairwise.neifst
- “WC84” see pairwise.WCfst

Value

A matrix of pairwise genetic distance
Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

References


See Also
pairwise.WCfstat pairwise.neifst

Examples

data(gtrunchier)
genet.dist(gtrunchier[,,-1])
genet.dist(gtrunchier[,,-1],method="Dr")

---

Genind2hierfstat

Converts genind objects from adegenet into a hierfstat data frame

Description
Converting genind objects from adegenet into a hierfstat data frame

Usage
genind2hierfstat(dat, pop=NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>a genind object</td>
</tr>
<tr>
<td>pop</td>
<td>a vector containing the population to which each individual belongs. If pop=NULL, pop taken from slot pop of the genind object</td>
</tr>
</tbody>
</table>

Value

a data frame with nloci+1 columns and ninds rows. The first column contains the population identifier, the following the genotypes at each locus
## genot2al

### Examples

```r
## Not run:
library(adegenet)
data(nancycats)
genind2hierfstat(nancycats)
basic.stats(nancycats)
genet.dist(nancycats)
data(H3N2)
basic.stats(genind2hierfstat(H3N2, pop=rep(1, dim(H3N2@tab)[1])), diploid=FALSE)

## End(Not run)
```

---

**genot2al**

Separates diploid genotypes in its constituent alleles

### Description

Separates the input vector of diploid genotypes in two vectors each containing one allele, and returns a vector of length $2 \times \text{length}(y)$ with the second part being the second allele.

### Usage

```r
genot2al(y)
```

### Arguments

- `y`  
  the diploid genotypes at one locus

### Value

returns a vector of length $2 \times \text{length}(y)$, with the second half of the vector containing the second alleles

### Author(s)

Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland

<jerome.goudet@unil.ch>

### References


### See Also

`varcomp`
Examples

data(gtrunchier)
genot2al(gtrunchier[,4])

default

getal

Converts diploid genotypic data into allelic data

Description

Converts diploid genotypic data into allelic data

Usage

getal(data)

Arguments

data

A data frame where the first column contains the population to which the different individuals belong, and the following columns contain the genotype of the individuals -one locus per column-

Value

data.al

A new data frame, with twice as many rows as the input data frame and one extra column. Each row of the first half of the data frame contains the first allele for each locus, and each row of the second half of the data frame contains the second allele at the locus. The extra column in second position corresponds to the identifier of the individual to which the allele belongs.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(gtrunchier)
getal(data.frame(gtrunchier[,2]))
Converts diploid genotypic data into allelic data

Description
Converts a data frame of genotypic diploid data with as many lines as individuals (ni) and as many columns as loci (nl) into an array [ni,nl,2] of allelic data

Usage
getal.b(data)

Arguments
- data: a data frame with ni rows and nl columns. Each line encodes one individual, each column contains the genotype at one locus of the individual

Value
an array [ni,nl,2] of alleles. The two alleles are stored in the third dimension of the array

Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

Examples
```
data(gtrunchier)
#multilocus diploid genotype of the first individual
gtrunchier[,,-c(1:2)]
#the diploid genotype splitted in its two constituent alleles
getal.b(gtrunchier[,,-c(1:2)][1,,]
```

Converts a Genetic Relationship Matrix (GRM) to a kinship matrix

Description
Converts a Genetic Relationship Matrix (GRM) to a kinship matrix

Usage
grm2kinship(x)
Arguments

x a square (GRM) matrix

Details

\[ k[ii] = x[ii] - 1; k[ij] = x[ij]/2 \]

Value

a kinship matrix

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

---

gtrunchier Genotypes at 6 microsatellite loci of Galba truncatula from different patches in Western Switzerland

Description

Data set consisting of the microsatellite genotypes of 370 Galba truncatula, a tiny freshwater snail, collecting from different localities and several patches within localities in Western Switzerland.

Usage

data(gtrunchier)

Value

<table>
<thead>
<tr>
<th>Localilty</th>
<th>Identifier of the locality of origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patch</td>
<td>Identifier of the patch of origin</td>
</tr>
<tr>
<td>L21.V</td>
<td>Genotype at locus L21.V. For instance the first individual carries allele 2 and 2 at this locus</td>
</tr>
<tr>
<td>L37.J</td>
<td>Genotype at locus L37.J</td>
</tr>
<tr>
<td>L29.V</td>
<td>Genotype at locus L29.V</td>
</tr>
</tbody>
</table>

References


hierfstat  General information on the hierfstat package

Description

This package contains functions to estimate hierarchical F-statistics for any number of hierarchical levels using the method described in Yang (1998). It also contains functions allowing to test the significance of population differentiation at any given level using the likelihood ratio G-statistic, showed previously to be the most powerful statistic to test for differentiation (Goudet et al., 1996). The difficulty in a hierarchical design is to identify which units should be permuted. Functions samp.within and samp.between give permutations of a sequence that allows reordering of the observations in the original data frame. An exemple of application is given in the help page for function g.stats.glob.

Hierfstat includes now all the capabilities of Fstat, and many others. A new serie of functions implementing the statistics described in Weir and Goudet (2017) and Goudet et al. (2018) (beta.dosage, fs.dosage) have been written to deal with large genomic data sets and take as input a matrix of allelic dosages, the number of alternate alleles an individual carries at a locus.

Several functions have been written to simulate genetic data, or to import them from existing sofwares such as quantiNemo or Hudson’s ms

Hierfstat links easily with the gaston, SNPRelate and adegenet packages, among others.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

References


### ind.count

**individual counts**

**Description**

Counts the number of individual genotyped per locus and population.

**Usage**

```r
ind.count(data)
```

**Arguments**

- `data` a data frame containing the population of origin in the first column and the genotypes in the following ones.

**Value**

A table –with np (number of populations) columns and nl (number of loci) rows—of genotype counts.

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**Examples**

```r
data(gtrunchier)
ind.count(gtrunchier[,-2])
```

### indpca

**PCA on a matrix of individuals genotypes frequencies**

**Description**

Carry out a PCA on the centered, unscaled matrix of individual’s allele frequencies.

**Usage**

```r
indpca(dat,ind.labels=NULL,scale=FALSE)
```

**Print Method**

```r
## S3 method for class 'indpca'
print(x,...)
```

**Plot Method**

```r
## S3 method for class 'indpca'
plot(x,eigen=FALSE,ax1=1,ax2=2,...)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>A data frame with population of origin as first column, and genotypes in following columns.</td>
</tr>
<tr>
<td>ind.labels</td>
<td>a vector of labels for the different individuals</td>
</tr>
<tr>
<td>scale</td>
<td>whether to standardize each column to variance 1 or to leave it as is (default)</td>
</tr>
<tr>
<td>x</td>
<td>an indpca object</td>
</tr>
<tr>
<td>eigen</td>
<td>whether to plot in an additional windows screeplot of the inertias for the different axes</td>
</tr>
<tr>
<td>ax1</td>
<td>which PCA coordinates to plot on the x axis</td>
</tr>
<tr>
<td>ax2</td>
<td>which PCA coordinates to plot on the y axis</td>
</tr>
<tr>
<td>...</td>
<td>further arguments to pass to print or plot</td>
</tr>
</tbody>
</table>

Value

An object of class `indpca` with components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>The function call</td>
</tr>
<tr>
<td>ipca</td>
<td>an object of class pca and dudi (see dudi.pca) in package ade4</td>
</tr>
<tr>
<td>mati</td>
<td>the original non centered matrix of individuals X alleles frequencies</td>
</tr>
</tbody>
</table>

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

```r
## not run
data(gtrunchier)
x<-indpca(gtrunchier[,-2],ind.labels=gtrunchier[,2])
plot(x,col=gtrunchier[,1],cex=0.7)
```

---

**kinship2dist**  
*Converts a kinship matrix to a distance matrix*

Description

Converts a kinship matrix to a distance matrix

Usage

```r
kinship2dist(x)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A square matrix containg kinship coefficients</td>
</tr>
</tbody>
</table>
Details

\[ D_{ij} = 0, D_{ij} = \frac{1-(x_{-\min(x)})}{(1-\min(x))} \]

Value

A distance matrix

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

---

**kinship2grm**

Converts a kinship matrix to a Genetic Relation Matrix (GRM)

Description

Converts a kinship matrix to a Genetic Relation Matrix (GRM)

Usage

`kinship2grm(x)`

Arguments

- `x` a square matrix containing kinship coefficients

Details

for off-diagonal elements, \( GRM = 2 \times x_{ij} \); for diagonal elements, \( GRM = 1 + x_{ii} \)

Value

a GRM matrix

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

```r
## Not run:
dos<-matrix(sample(0:2,replace=TRUE,size=1000),nrow=10) #dosage matrix for 10 inds at 100 loci
ks<-beta.dosage(dos) # kinship matrix
kinship2grm(ks)
## End(Not run)
```
kinshipShift

Description

Shifts a kinship matrix

Usage

kinshipShift(x, shift=NULL)

Arguments

- **x**: a square matrix
- **shift**: the amount by which the elements of x should be shifted. If shift=NULL, the average of the off-diagonal elements is substracted

Details

The kinship matrix produced by beta.dosage is relative to the average kinship of the set of individuals analysed \((1/(n(n-1)/2) \sum_i \sum_{j>i} x_{ij} = 0)\). Another reference point might be useful, for instance to avoid negative kinship values, one might want to shift the matrix by \(\min(x_{ij}), i \neq j\).

Value

The shifted kinship matrix \(\frac{x_{ij} - shift}{1-shift}\)

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

mat2vec

Description

creates a vector from a matrix

Usage

mat2vec(mat, upper=FALSE)

Arguments

- **mat**: a symmetric matrix
- **upper**: whether the upper triangular matrix is to be copied to the vector
Value

a vector

Examples

{
  mat2vec(matrix(1:16,nrow=4))
  mat2vec(matrix(1:16,nrow=4),upper=TRUE)
}

ms2bed

Import the output of the ms program in a BED object

Description

Import the output of the ms program into a BED object, as defined in the gaston package

Usage

ms2bed(fname)

Arguments

fname the name of the text file containing ms output

Value

a bed object

ms2dos

Import ms output

Description

Import the output of the ms program into suitable format for further manipulation

Usage

ms2dos(fname)

Arguments

fname a text file containing the output of the ms program
**Value**

alldat a matrix with as many row as (haploid) individuals and as many columns as SNPs

bim a data frame with two components chr contains the chromosome (replicate) id; pos contains the SNPs position on the chromosome

---

<table>
<thead>
<tr>
<th>nb.alleles</th>
<th>Number of different alleles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Counts the number of different alleles at each locus and population

**Usage**

`nb.alleles(data, diploid=TRUE)`

**Arguments**

- **data** A data frame containing the population of origin in the first column and the genotypes in the following ones
- **diploid** whether individuals are diploid

**Value**

A table, –with np (number of populations) columns and nl (number of loci) rows– of the number of different alleles

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**Examples**

```r
data(gtrunchier)
nb.alleles(gtrunchier[, -2])
```
pairwise.neifst  Estimates pairwise FSTs according to Nei (1987)

Description
Estimate pairwise FSTs according to Nei (1987)

Usage
pairwise.neifst(dat, diploid=TRUE)

Arguments
dat        A data frame containing population of origin as the first column and multi-locus genotypes in following columns
diploid    whether the data is from a diploid (default) or haploid organism

Details
FST are calculated using Nei (87) equations for FST’, as described in the note section of basic.stats

Value
A matrix of pairwise FSTs

Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

References

See Also
pairwise.WCfst genet.dist

Examples
data(gtrunchier)
pairwise.neifst(gtrunchier[-2], diploid=TRUE)
Estimates pairwise FSTs according to Weir and Cockerham (1984)

**Description**

Estimates pairwise FSTs according to Weir and Cockerham (1984)

**Usage**

```r
pairwise.WCfst(dat, diploid = TRUE)
```

**Arguments**

- `dat`: A data frame containing population of origin as the first column and multi-locus genotypes in following columns
- `diploid`: whether the data is from a diploid (default) or haploid organism

**Details**

FST are calculated using Weir & Cockerham (1984) equations for FST', as described in the note section of `wc`

**Value**

A matrix of pairwise FSTs

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**References**


**See Also**

- `pairwise.neifst`
- `genet.dist`

**Examples**

```r
data(gtrunchier)
pairwise.WCfst(gtrunchier[, -2], diploid = TRUE)
```
pcoa

Principal coordinate analysis

Description

principal coordinates analysis as described in Legendre & Legendre Numerical Ecology

Usage

pcoa(mat,plotit=TRUE,...)

Arguments

mat a distance matrix
plotit Whether to produce a plot of the pcoa
... further arguments (graphical for instance) to pass to the function

Value

valp the eigen values of the pcoa
vecp the eigen vectors of the pcoa (the coordinates of observations)
euc1 The cumulative euclidian distances among observations,

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(gtrunchier)
colo<-c("black","red","blue","yellow","orange","green")
pcoa(as.matrix(genet.dist(gtrunchier[,-1])),col=rep(colo,c(5,5,4,5,5,5)))

pop.freq

Allelic frequencies

Description

Estimates allelic frequencies for each population and locus

Usage

pop.freq(data,diploid=TRUE)
Arguments

data a data frame where the first column contains the population to which the different individuals belong, and the following columns contain the genotype of the individuals -one locus per column-
diploid specify whether the data set consists of diploid (default) or haploid data

Value

A list containing allele frequencies. Each element of the list is one locus. For each locus, Populations are in columns and alleles in rows

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(gtrunchier)
pop.freq(gtrunchier[, -2])

pp.fst

fst per pair

Description

fst per pair following Weir and Cockerham (1984)

Usage

pp.fst(dat = dat, diploid = TRUE, ...)

Arguments

dat a genetic data frame
diploid whether data from diploid organism
... further arguments to pass to the function

Value

call function call
fst.pp pairwise Fsts
vc.per.loc for each pair of population, the variance components per locus

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>
References


---

**pp.sigma.loc**

wrapper to return per locus variance components

---

**Description**

wrapper to return per locus variance components between pairs of samples x & y

**Usage**

```r
pp.sigma.loc(x,y,dat=dat,diploid=TRUE,...)
```

**Arguments**

- **x, y**  samples 1 and 2
- **dat**  a genetic data set
- **diploid**  whether dat are diploid
- **...**  further arguments to pass to the function

**Value**

- **sigma.loc**  variance components per locus

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

---

**print.pp.fst**

print function for pp.fst

---

**Description**

print function for pp.fst

**Usage**

```r
## S3 method for class 'pp.fst'
print(x,...)
```
Arguments

- `x` an object of class `pp.fst`
- `...` further arguments to pass to the function

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Description

Read QuantiNemo extended format for genotype files

Read QuantiNemo (http://www2.unil.ch/popgen/softwares/quantinemo/) genotype files extended format (option 2)

Usage

```r
qn2.read.fstat(fname, na.s = c("NA","NaN"))
```

Arguments

- `fname` quantinemo file name
- `na.s` na string used

Value

- `data` data frame with `ntsb+1` columns, the first being the population to which the individual belongs and the next being the genotypes, one column per locus; and `ninds` rows
- `sex` the sex of the individuals

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

References


See Also

read.fstat

Examples

```r
dat<-qn2.read.fstat(system.file("extdata","qn2_sex.dat",package="hierfstat"))
sexbias.test(dat[[1]],sex=dat[[2]])
```

---

**read.fstat**  
**Reads data from a FSTAT file**

### Description

Imports a FSTAT data file into R. The data frame created is made of \( n_l+1 \) columns, \( n_l \) being the number of loci. The first column corresponds to the Population identifier, the following columns contain the genotypes of the individuals.

### Usage

```r
read.fstat(fname, na.s = c("0","00","000","0000","00000","000000","NA"))
```

### Arguments

- **fname**: a file in the FSTAT format ([http://www.unil.ch/popgen/softwares/fstat.htm](http://www.unil.ch/popgen/softwares/fstat.htm)): The file must have the following format:
  - The first line contains 4 numbers: the number of samples, \( n_p \), the number of loci, \( n_l \), the highest number used to label an allele, \( n_u \), and a 1 if the code for alleles is a one digit number (1-9), a 2 if code for alleles is a 2 digit number (01-99) or a 3 if code for alleles is a 3 digit number (001-999). These 4 numbers need to be separated by any number of spaces.
  - The first line is immediately followed by \( n_l \) lines, each containing the name of a locus, in the order they will appear in the rest of the file.
  - On line \( n_l+2 \), a series of numbers as follow:
    ```
    1 0102 0103 0101 0203 0 0303
    ```
    - The first number identifies the sample to which the individual belongs, the second is the genotype of the individual at the first locus, coded with a 2 digits number for each allele, the third is the genotype at the second locus, until locus \( n_l \) is entered (in the example above, \( n_l=6 \)). Missing genotypes are encoded with 0, 00, 0000, 000000 or NA. Note that 0001 or 0100 are not a valid format, as both alleles at a locus have to be known, otherwise, the genotype is considered as missing. No empty lines are needed between samples.

- **na.s**: The strings that correspond to the missing value. *You should note have to change this*
read.fstat.data

Value

a data frame containing the desired data, in a format adequate to pass to varcomp

References


Examples

read.fstat(paste(path.package("hierfstat"),"/extdata/diploid.dat",sep="",collapse=""))

read.fstat.data

Reads data from a FSTAT file

Description

Imports a FSTAT data file into R. The data frame created is made of nl+1 columns, nl being the number of loci. The first column corresponds to the Population identifier, the following columns contains the genotypes of the individuals.

Usage

read.fstat.data(fname, na.s = c("0","00","000","0000","00000","000000","NA"))

Arguments

c

fname a file in the FSTAT format: The file must have the following format:
The first line contains 4 numbers: the number of samples, np , the number of loci, nl, the highest number used to label an allele, nu, and a 1 if the code for alleles is a one digit number (1-9), a 2 if code for alleles is a 2 digit number (01-99) or a 3 if code for alleles is a 3 digit number (001-999). These 4 numbers need to be separated by any number of spaces.
The first line is immediately followed by nl lines, each containing the name of a locus, in the order they will appear in the rest of the file.
On line nl+2, a series of numbers as follow:
1 0102 0103 0101 0203 0 0303
The first number identifies the sample to which the individual belongs, the second is the genotype of the individual at the first locus, coded with a 2 digits number for each allele, the third is the genotype at the second locus, until locus nl is entered (in the example above, nl=6). Missing genotypes are encoded with 0, 00, 0000, 000000 or NA. Note that 0001 or 0100 are not a valid format, as both alleles at a locus have to be known, otherwise, the genotype is considered as missing. No empty lines are needed between samples.
The strings that correspond to the missing value. *You should note have to change this*

Value

a data frame containing the desired data, in a format adequate to pass to `varcomp`

References


Examples

```r
read.fstat.data(paste(path.package("hierfstat"),"/extdata/diploid.dat",sep="",collapse=""))
```

---

**Description**

With argument `what="SNP"`, each site is read as a SNP, with the ancestral allele encoded as 0 and the alternate allele encoded as 1. If the ms output file contains several replicates, the different replicates will be collated together. Hence, the number of loci is the sum of all sites from all replicates.

**Usage**

```r
read.ms(fname,what=c("SNP","Haplotype"))
```

**Arguments**

- `fname` file name containing ms output
- `what` whether to read ms output as SNPs or haplotypes

**Details**

With argument `what="Haplotype"`, each different sequence from a replicate is read as a haplotype, by converting it first to a factor, and then to an integer. There will be as many loci as there are replicates, and the number of alleles per locus will be the number of different haplotypes in the corresponding replicate.

**Value**

`alldat` a data frame with `nloc+1` columns, the first being the population to which the individual belongs and the next being the genotypes, one column per locus; and one row per (haploid) individual.
read.VCF

Reads a VCF file into a BED object

Description

Reads a [https://samtools.github.io/hts-specs/](https://samtools.github.io/hts-specs/) Variant Call Format (VCF) file into a BED object, retaining bi-allelic SNPs only

Usage

```
read.VCF(fname,BiAllelic=TRUE,...)
```

Arguments

- `fname` VCF file name. The VCF file can be compressed (VCF.gz)
- `BiAllelic` Logical. If TRUE, only bi-allelic SNPs are retained, otherwise, all variant are kept
- `...` other arguments to pass to the function

Value

A `bed.matrix-class` object

See Also

`read.vcf`
Examples

```r
filepath <- system.file("extdata", "LCT.vcf.gz", package="gaston")
x1 <- read.VCF( filepath )
x1
```

---

### samp.between

**Shuffles a sequence among groups defined by the input vector**

#### Description

Used to generate a permutation of a sequence `1:length(lev)`. blocks of observations are permuted, according to the vector `lev` passed to the function.

#### Usage

```r
samp.between(lev)
```

#### Arguments

- `lev` a vector containing the groups to be permuted.

#### Value

A vector `1:length(lev)` (with blocks defined by data) randomly permuted. Usually, one passes the result to reorder observations in a data set in order to carry out permutation-based tests.

#### Author(s)

Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland

<jerome.goudet@unil.ch>

#### References


#### See Also

- `samp.within`
- `g.stats.glob`

#### Examples

```r
samp.between(rep(1:4,each=4))
# for an application see example in g.stats.glob
```
samp.between.within  Shuffles a sequence

Description

Used to generate a permutation of a sequence 1:length(inner.lev). blocks of observations defined by inner.lev are permuted within blocks defined by outer.lev

Usage

samp.between.within(inner.lev, outer.lev)

Arguments

inner.lev  a vector containing the groups to be permuted.
outer.lev  a vector containing the blocks within which observations are to be kept.

Value

a vector 1:length(lev) (with blocks defined by data) randomly permuted. Usually, one passes the result to reorder observations in a data set in order to carry out permutation-based tests

See Also

test.between.within.

samp.within  Shuffles a sequence within groups defined by the input vector

Description

Used to generate a permutation of a sequence 1:length(lev). observations are permuted within blocks, according to the vector lev passed to the function.

Usage

samp.within(lev)

Arguments

lev  a vector containing the group to which belongs the observations to be permuted.
Value

a vector 1:length(lev) (with blocks defined by

lev

) randomly permuted. Usually, one passes the result to reorder observations in a data set in order to carry out permutation-based tests.

Author(s)

Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland
<jerome.goudet@unil.ch>

References


See Also

samp.between,g.stats.glob.

Examples

samp.within(rep(1:4,each=4))
#for an application see example in g.stats.glob

sexbias.test  Test for sex biased dispersal

Description

Test whether one sex disperses more than the other using the method described in Goudet etal. (2002)

Usage

sexbias.test(dat,sex,nperm=NULL,test="mAIC",alternative="two.sided")

Arguments

dat a data frame with n.locs+1 columns and n.ind rows
sex a vector containing the individual’s sex
nperm the number of permutation to carry out
test one of "mAIC" (default), "vAIC","FIS" or "FST"
alternative one of "two.sided" (default),"less" or "greater"
\textbf{Value}

- call the function call
- res the observation for each sex
- statistic the observed statistic for the chosen test
- p.value the p-value of the hypothesis

\textbf{Author(s)}

Jerome Goudet \textless jerome.goudet@unil.ch\textgreater

\textbf{References}


\textbf{Examples}

\begin{verbatim}
data(crocrussula)
sexbias.test(crocrussula$genot, crocrussula$sex)
dat<-qn2.read.fstat(system.file("extdata","qn2_sex.dat",package="hierfstat"))
sexbias.test(dat[[1]], sex=dat[[2]])
## Not run:
sexbias.test(crocrussula$genot, crocrussula$sex, nperm=1000)
sexbias.test(dat[[1]], sex=dat[[2]], nperm=100, test="FST", alternative="greater")
## End(Not run)
\end{verbatim}

\textbf{sim.freq}

\textit{Simulates frequencies, for internal use only}

\textbf{Description}

Simulates frequencies, for internal use only

\textbf{sim.genot}

\textit{Simulates genotypes in an island model at equilibrium}

\textbf{Description}

Simulates genotypes from several individuals in several populations at several loci in an island model at equilibrium. The islands may differ in size and inbreeding coefficients.

\textbf{Usage}

\begin{verbatim}
sim.genot(size=50, nbal=4, nbloc=5, nbpop=3, N=1000, mig=0.001, mut=0.0001, f=0)
\end{verbatim}
Arguments

- **size**: The number of individuals to sample per population
- **nbal**: The maximum number of alleles present at a locus
- **nbloc**: The number of loci to simulate
- **nbpop**: The number of populations to simulate
- **N**: The population sizes for each island
- **mig**: the proportion of migration among islands
- **mut**: The loci mutation rate
- **f**: the inbreeding coefficient for each island

Value

A data frame with nbpop*size lines and nbloc+1 columns. Individuals are in rows and genotypes in columns, the first column being the population identifier.

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

```r
## Not run:
dat<-sim.genot(nbpop=4,nbal=20,nbloc=10,mig=0.001,mut=0.0001,N=c(100,100,1000,1000),f=0)
betas(dat)$betaiovl
## End(Not run)
```

Description

This function allows to simulate genetic data from a metapopulation model, where each population can have a different size and a different inbreeding coefficient, and migration between each population is given in a migration matrix.

This function simulates genetic data under a migration matrix model. Each population $i$ sends a proportion of migrant alleles $m_{ij}$ to population $j$ and receives a proportion of migrant alleles $m_{ji}$ from population $j$.

Usage

```r
sim.genot.metapop.t(size=50,nbal=4,nbloc=5,nbpop=3,N=1000,
mig=diag(3),mut=0.0001,f=0,t=100)
```
**Arguments**

- `size`: the number of sampled individuals per population
- `nbal`: the number of alleles per locus (maximum of 99)
- `nbloc`: the number of loci to simulate
- `nbpop`: the number of populations to simulate
- `N`: the effective population sizes of each population. If only one number, all populations are assumed to be of the same size
- `mig`: a matrix with `nbpop` rows and columns giving the migration rate from population `i` (in row) to population `j` (in column). Each row must sum to 1.
- `mut`: the mutation rate of the loci
- `f`: the inbreeding coefficient for each population
- `t`: the number of generation since the islands were created

**Details**

In this model, $\theta_t$ can be written as a function of population size $N_i$, migration rate $m_{ij}$, mutation rate $\mu$ and $\theta_{(t-1)}$.

The rational is as follows:

With probability $\frac{1}{N_i}$, 2 alleles from 2 different individuals in the current generation are sampled from the same individual of the previous generation:

- Half the time, the same allele is drawn from the parent;
- The other half, two different alleles are drawn, but they are identical in proportion $\theta_{(t-1)}$.

- With probability $1 - \frac{1}{N_i}$, the 2 alleles are drawn from different individuals in the previous generation, in which case they are identical in proportion $\theta_{(t-1)}$.

This holds providing that neither alleles have mutated or migrated. This is the case with probability $m_{ii}^2 \times (1 - \mu)^2$. If an allele is a mutant, then its coancestry with another allele is 0.

Note also that the mutation scheme assumed is the infinite allele (or site) model. If the number of alleles is finite (as will be the case in what follows), the corresponding mutation model is the K-allele model and the mutation rate has to be adjusted to $\mu' = \frac{K-1}{K} \mu$.

Continue derivation

**Value**

A data frame with `size*nbpop` rows and `nbloc+1` columns. Each row is an individual, the first column contains the identifier of the population to which the individual belongs, the following `nbloc` columns contain the genotype for each locus.

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>
Examples

#2 populations
psize<-c(10,1000)
mig.mat<-matrix(c(0.99,0.01,0.1,0.9),nrow=2,byrow=TRUE)
dat<-sim.genot.metapop.t(nbal=10,nbloc=100,nbpop=2,N=psize,mig=mig.mat,mut=0.00001,t=100)
betas(dat)$betaiovl # Population specific estimator of FST

#1D stepping stone
## Not run:
np<-10
m<-0.2
mig.mat<-diag(np)*(1-m)
diag(mig.mat[-1,-np])<-m/2
diag(mig.mat[-np,-1])<-m/2
mig.mat[1,1:2]<-c(1-m/2,m/2)
mig.mat[np,(np-1):np]<-c(m/2,1-m/2)
dat<-sim.genot.metapop.t(nbal=10,nbloc=50,nbpop=np,mig=mig.mat,t=400)
pcoa(as.matrix(genet.dist(dat))) # principal coordinates plot

## End(Not run)

---

sim.genot.t

Simulate data from a non equilibrium continent-island model

Description

This function allows to simulate genetic data from a non-equilibrium continent-island model, where each island can have a different size and a different inbreeding coefficient.

This function simulates genetic data under the continent-islands model (IIM=TRUE) or the finite island model (IIM=FALSE). In the IIM, a continent of infinite size sends migrants to islands of finite sizes $N_i$ at a rate $m$. Alleles can also mutate to a new state at a rate $\mu$. Under this model, the expected $F_{ST,i}$, $\theta_i$, can be calculated and compared to empirical estimates.

Usage

sim.genot.t(size=50,nbal=4,nbloc=5,nbpop=3,N=1000,
mig=0.001,mut=0.0001,f=0,t=100,IIM=TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>the number of sampled individuals per island</td>
</tr>
<tr>
<td>nbal</td>
<td>the number of alleles per locus (maximum of 99)</td>
</tr>
<tr>
<td>nbloc</td>
<td>the number of loci to simulate</td>
</tr>
<tr>
<td>nbpop</td>
<td>the number of islands to simulate</td>
</tr>
</tbody>
</table>
The effective population sizes of each island. If only one number, all islands are assumed to be of the same size.

The migration rate from the continent to the islands.

The mutation rate of the loci.

The inbreeding coefficient for each island.

The number of generation since the islands were created.

Whether to simulate a continent island Model (default) or a migrant pool island Model.

In this model, $\theta_t$ can be written as a function of population size $N_i$, migration rate $m$, mutation rate $\mu$ and $\theta_{(t-1)}$.

The rational is as follows:

With probability $\frac{1}{N}$, 2 alleles from 2 different individuals in the current generation are sampled from the same individual of the previous generation:

- Half the time, the same allele is drawn from the parent;
- The other half, two different alleles are drawn, but they are identical in proportion $\theta_{(t-1)}$.

With probability $1-\frac{1}{N}$, the 2 alleles are drawn from different individuals in the previous generation, in which case they are identical in proportion $\theta_{(t-1)}$.

This holds providing that neither alleles have mutated or migrated. This is the case with probability $(1-m)^2 \times (1-\mu)^2$. If an allele is a mutant or a migrant, then its coancestry with another allele is 0 in the infinite continent-islands model (it is not the case in the finite island model).

Note also that the mutation scheme assumed is the infinite allele (or site) model. If the number of alleles is finite (as will be the case in what follows), the corresponding mutation model is the K-allele model and the mutation rate has to be adjusted to $\mu' = \frac{K-1}{K} \mu$.

Let's substitute $\alpha$ for $(1-m)^2 (1-\mu)^2$ and $x$ for $\frac{1}{2N}$.

The expectation of $F_{ST}$, $\theta$ can be written as:

$$
\theta_t = (\alpha (1-x))^t \theta_0 + \frac{x}{1-x} \sum_{i=1}^{t} (\alpha (1-x))^i
$$

which reduces to $\theta_t = \frac{x}{1-x} \sum_{i=1}^{t} (\alpha (1-x))^i$ if $\theta_0 = 0$.

Transition equations for $theta$ in the migrant-pool island model (IIM=FALSE) are given in Rouseet (1996). Currently, the migrant pool is made of equal contribution from each island, irrespective of their size.

A data frame with size*nbpop rows and nbloc+1 columns. Each row is an individual, the first column contains the island to which the individual belongs, the following nbloc columns contain the genotype for each locus.
Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

References

Examples

```r
psize<-c(100,1000,10000,100000,1000000)
dat<-sim.genot.t(nbal=4,nbloc=20,nbpop=5,N=psize,mig=0.001,mut=0.0001,t=100)

summary(wc(dat)) #Weir and cockerham overall estimators of FST & FIS
betas(dat) # Population specific estimator of FST
```

---

**subsampind**

Subsample a FSTAT data frame

Description
Subsample a given number of individuals from a FSTAT data frame

Usage
`subsampind(dat,sampsize = 10)`

Arguments
dat A data frame with population of origin as first column, and genotypes in following columns.
sampsize the number of individuals to sample in each population.

Value
A data frame with population of origin as first column, and genotypes in following columns. Each population is made of at most sampsize individuals

Author(s)
Jerome Goudet <jerome.goudet@unil.ch>

Examples
```r
data(gtrunchier)
subsampind(gtrunchier[,,-1],6) # check the warning
```
test.between

Tests the significance of the effect of test.lev on genetic differentiation

Description

Tests the significance of the effect of test.lev on genetic differentiation

Usage

test.between(data, test.lev, rand.unit, nperm, ...)

Arguments

data a data frame containing the genotypes for the different loci
test.lev A vector containing the units from which to construct the contingency tables
rand.unit A vector containing the assignment of each observation to the units to be permuted
nperm The number of permutations to carry out for the test
... Mainly here to allow passing diploid=FALSE if necessary

Value

g.star A vector containing all the generated g-statistics, the last one being the observed
p.val The p-value associated with the test

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(gtrunchier)
attach(gtrunchier)
# test whether the locality level has a significant effect on genetic structuring
test.between(gtrunchier[, -c(1,2)], test.lev=Locality, rand.unit=Patch)
**test.between.within**  
Tests the significance of the effect of test.lev on genetic differentiation

Description

Tests, using permutations of rand.unit within units defined by the vector within the significance of the contingency tables allele X (levels of test.lev)

Usage

test.between.within(data, within, test.lev, rand.unit, nperm, ...)

Arguments

data  
a data frame containing the genotypes for the different loci

within  
A vector containing the units in which to keep the observations

test.lev  
A vector containing the units from which to construct the contingency tables

rand.unit  
A vector containing the assignment of each observation to the units to be permuted

nperm  
The number of permutations to carry out for the test

...  
Mainly here to allow passing diploid=FALSE if necessary

Value

g.star  
A vector containing all the generated g-statistics, the last one being the observed

p.val  
The p-value associated with the test

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(yangex)
attach(yangex)
#tests for the effect of spop on genetic structure
test.between.within(data.frame(genot),within=pop,test=spop,rand=sspop)
test.g  \hspace{1cm} Tests the significance of the effect of level on genetic differentiation

Description

Tests the significance of the effect of level on genetic differentiation

Usage

test.g(data = data, level, nperm = 100, ...)

Arguments

data \hspace{1cm} a data frame containing the genotypes for the different loci
level \hspace{1cm} A vector containing the assignment of each observation to its level
nperm \hspace{1cm} The number of permutations to carry out for the test
... \hspace{1cm} Mainly here to allow passing diploid=FALSE if necessary

Value

g.star \hspace{1cm} A vector containing all the generated g-statistics, the last one being the observed
p.val \hspace{1cm} The p-value associated with the test

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

data(gtrunchier)
attach(gtrunchier)
test.g(gtrunchier[-c(1,2),Locality]

---

test.within  \hspace{1cm} Tests the significance of the effect of inner.level on genetic differentiation within blocks defined by outer.level

Description

Tests the significance of the effect of inner.level on genetic differentiation within blocks defined by outer.level

Usage

test.within(data, within, test.lev, nperm, ...)

---

Arguments

- **data**
  a data frame containing the genotypes for the different loci
- **within**
  A vector containing the units in which to keep the observations
- **test.lev**
  A vector containing the units from which to construct the contingency tables
- **nperm**
  The number of permutations to carry out for the test
- **...**
  Mainly here to allow passing `diploid=FALSE` if necessary

Value

- **g.star**
  A vector containing all the generated g-statistics, the last one being the observed
- **p.val**
  The p-value associated with the test

Author(s)

Jerome Goudet <jerome.goudet@unil.ch>

Examples

```r
data(gtrunchier)
attach(gtrunchier)
#tests whether the patch level has a significant effect on genetic structure
test.within(gtrunchier[, -c(1,2)], within=Locality, test.lev=Patch)
```

---

**varcomp**

*Estimates variance components for each allele of a locus*

Description

Estimates variance components for each allele for a (fully) hierarchical random design defined by all but the last column of the data frame `data`, the last column containing the genetic data to analyse. Columns for the hierarchical design should be given from the outermost to the innermost before the individual (e.g. continent, region, population, patch,...)

Usage

```r
varcomp(data, diploid=TRUE)
```

Arguments

- **data**
  a data frame that contains the different factors from the outermost (e.g. region) to the innermost before the individual. The last column of the data frame `data` contains the locus to analyse, which can be multiallelic. Missing data are allowed.
- **diploid**
  a boolean stating whether the data come from diploid (TRUE=default) or haploid (FALSE) organisms
Details

The format for genotypes is simply the code for the 2 alleles put one behind the other, without space in between. For instance if allele 1 at the locus has code 23 and allele 2 39, the genotype format is 2339.

Value

df the degrees of freedom for each level
k the k matrix, the coefficients associated with the variance components
res the variance components for each allele
overall the variance components summed over alleles
F a matrix of hierarchical F-statistics type-coefficients with the first line corresponding to $F_{(n-1)/n}, F_{(n-2)/n} \ldots F_{i/n}$ and the diagonal corresponding to $F_{(n-1)/n}, F_{(n-2)/(n-1)}, F_{i/2}$

Author(s)

Jerome Goudet, DEE, UNIL, CH-1015 Lausanne Switzerland

<jerome.goudet@unil.ch>

http://www.unil.ch/popgen/people/jerome.htm

References


See Also

varcomp.glob.

Examples

#load data set
data(gtrunchier)
attach(gtrunchier)
#
varcomp(data.frame(Locality,Patch,L21.V))
Estimate variance components and hierarchical F-statistics over all loci

Description

Return multilocus estimators of variance components and F-statistics

Usage

varcomp.glob(levels=levels,loci=loci,diploid=TRUE)

Arguments

- **levels**: a data frame containing the different levels (factors) from the outermost (e.g. region) to the innermost before the individual
- **loci**: a data frame containing the different loci
- **diploid**: Specify whether the data are coming from diploid or haploid organisms (diploid is the default)

Value

- **loc**: The variance components for each locus
- **overall**: The variance components summed over all loci
- **F**: a matrix of hierarchical F-statistics type-coefficients with the first line corresponding to $F_{(n-1)/n}, F_{(n-2)/n} ... F_i/n$ and the diagonal corresponding to $F_{(n-1)/n}, F_{(n-2)/(n-1)} F_i/2$

Author(s)

Jerome Goudet DEE, UNIL, CH-1015 Lausanne Switzerland

<jerome.goudet@unil.ch>

References


See Also

varcomp.
vec2mat

Examples

# load data set
data(gtrunchier)
attach(gtrunchier)
varcomp.glob(data.frame(Locality,Patch),gtrunchier[,,-c(1,2)])

vec2mat  

Fills a triangular matrix from the inputed vector

Description

Fills a triangular matrix from the inputed vector

Usage

vec2mat(x,diag=FALSE,upper=FALSE)

Arguments

x  
a vector
diag  
whether the vector contains the diagonal elements
upper  
whether the vector contains the upper triangular matrix elements

Value

a matrix

Examples

{
  vec2mat(1:10)
  vec2mat(1:10,diag=TRUE)
  vec2mat(1:10,upper=TRUE)
}
Computes Weir and Cockerham estimates of Fstatistics

**Usage**

```r
wc(ndat, diploid=TRUE, pol=0.0)
```

## S3 method for class 'wc'

```r
print(x,...)
```

**Arguments**

- `ndat`: data frame with first column indicating population of origin and following representing loci
- `diploid`: Whether data are diploid
- `pol`: level of polymorphism requested for inclusion. Note used for now
- `x`: an object of class wc
- `...`: further arguments to pass to print.wc

**Value**

- `sigma`: variance components of allele frequencies for each allele, in the order among populations, among individuals within populations and within individuals
- `sigma.loc`: variance components per locus
- `per.al`: FST and FIS per allele
- `per.loc`: FST and FIS per locus
- `FST`: FST overall loci
- `FIS`: FIS overall loci

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**Examples**

```r
data(gtrunchier)
wc(gtrunchier[,-1])
```
write.bayescan  

**Write a bayescan file**

**Description**
write the genotypes in a format suitable for analysis with bayescan

**Usage**

```r
write.bayescan(dat=dat,diploid=TRUE,fn="dat.bsc")
```

**Arguments**

- `dat`: a genotype data frame
- `diploid`: whether the dataset is diploid or haploid
- `fn`: file name for output

**Value**
a text file fn is written in the current directory

**Author(s)**
Jerome Goudet <jerome.goudet@unil.ch>

**References**
http://cmpg.unibe.ch/software/BayeScan/

write.fstat  

**Write an Fstat data file**

**Description**
Write a data frame to a text file in the fstat data format, see `read.fstat`

**Usage**

```r
write.fstat(dat,fname="genotypes.dat")
```

**Arguments**

- `dat`: A data frame with first column containing the population identifier and remaining columns containing genotypes
- `fname`: The name of the text file to which the data frame should be written
write.ped

Value
None

Author(s)
Jerome Goudet

References

Examples
```r
## Not run: data(gtrunchier)
write.fstat(gtrunchier[-1],"galba.dat")
## End(Not run)
```

---

write.ped

Write ped file for analyses with PLINK

Description
write a ped and a map file suitable for analysis with PLINK

Usage
```r
write.ped(dat, ilab = NULL, pop = NULL,
          fname = "dat",na.str="0",f.id=NULL,m.id=NULL,loc.pos=NULL,sex=NULL)
```

Arguments
- `dat` a hierfstat data frame
- `ilab` individual labels
- `pop` population id
- `fname` filename for ped file
- `na.str` character string to use for missing values
- `f.id` father id. default to unknown
- `m.id` mother id. default to unknown
- `loc.pos` the loci position default to unknown
- `sex` the individual sex. default to unknown
**write.struct**

**Value**

- a map file containing the loci positions
- a ped file containing genotypes etc...

**References**

- Chang et al. (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets

---

**Write structure file**

**Description**

Write a genotype data set to a file in the structure format

**Usage**

```r
write.struct(dat, ilab=NULL, pop=NULL, MARKER_NAMES=FALSE, MISSING=-9, fname="dat.str")
```

**Arguments**

- `dat`: a genotype dataframe
- `ilab`: whether to add a column with individual labels
- `pop`: whether to add a column with population identifiers
- `MARKER_NAMES`: whether to add a row with marker names
- `MISSING`: the code for missing alleles
- `fname`: a string containing the file name

**Value**

a text file in the structure format

**Author(s)**

Jerome Goudet <jerome.goudet@unil.ch>

**References**

Example data set from Yang (1998) appendix

Description

Reproduce the example data set used in Yang’s paper appendix. The genotype (column `genot`) is invented.

Usage

data(exhier)

Value

<table>
<thead>
<tr>
<th>pop</th>
<th>outermost level</th>
</tr>
</thead>
<tbody>
<tr>
<td>spop</td>
<td>sub pop level</td>
</tr>
<tr>
<td>sspop</td>
<td>sub sub pop level</td>
</tr>
<tr>
<td>genot</td>
<td>dummy diploid genotype</td>
</tr>
</tbody>
</table>

References


Examples

data(yangex)
varcomp(yangex)
#the k matrix should be the same as matrix (A2) in Yang's appendix, p. 956
Index

* IO
  write.fstat, 63

* datasets
  cont.isl, 14
  cont.isl99, 15
  crocrussula, 16
  diploid, 16
  exhier, 17
  gtrunchier, 28
  yangex, 66

* distance
  genet.dist, 23

* manip
  genot2al, 25
  getal, 26
  getal.b, 27
  read.fstat, 42
  read.fstat.data, 43
  samp.between, 46
  samp.between.within, 47
  samp.within, 47

* miscellaneous
  write.struct, 65

* misc
  hierfstat, 29

* multivariate
  indpca, 30

* nonparametric
  test.between, 55
  test.between.within, 56
  test.g, 57
  test.within, 57

* univar
  allele.count, 4
  allelic.richness, 5
  basic.stats, 6
  boot.ppffis, 11
  boot.vc, 13
  g.stats, 20
  g.stats.glob, 21
  ind.count, 30
  nb.alleles, 35
  pop.freq, 38
  pp.fst, 39
  pp.sigma.loc, 40
  print.pp.fst, 40
  varcomp, 58
  varcomp.glob, 60
  wc, 62

* utilities
  subsampind, 54

  AIC, 3
  allele.count, 4
  allelic.richness, 5

  basic.stats, 6, 36
  beta.dosage, 8
  betas, 9
  biall2dos, 11
  boot.ppffis, 11
  boot.ppffst, 12
  boot.vc, 13
  cont.isl, 14
  cont.isl99, 15
  crocrussula, 16

  diploid, 16
  exhier, 17

  fis.dosage (fs.dosage), 18
  fs.dosage, 18
  fst.dosage (fs.dosage), 18
  fstat2dos, 19

  g.stats, 20, 22
  g.stats.glob, 21, 21, 46, 48
  genet.dist, 23, 36, 37
genind2hierfstat, 24
getal, 25
getal, 26
getal.b, 27
grm2kinship, 27
gtruncate, 28
hierfstat, 29
ind.count, 8, 30
indpca, 30

kinship2dist, 31
kinship2grm, 32
kinshipShift, 33
mat2vec, 33
ms2bed, 34
ms2dos, 34

nb.alleles, 35

pairwise.neifst, 23, 24, 36, 37
pairwise.WCfst, 23, 24, 36, 37
pcoa, 38
plot.fs.dosage (fs.dosage), 18
plot.indpca (indpca), 30
pop.freq, 8, 38
pp.fst, 23, 39
pp.sigma.loc, 40
print.basic.stats (basic.stats), 6
print.boot.ppfst (boot.ppfst), 12
print.fs.dosage (fs.dosage), 18
print.indpca (indpca), 30
print.pp.fst, 40
print.wc (wc), 62
qn2.read.fstat, 41

read.fstat, 42, 42
read.fstat.data, 43
read.ms, 44
read.VCF, 45
read.vcf, 45

samp.between, 22, 46, 48
samp.between.within, 47
samp.within, 22, 46, 47
sexbias.test, 48
sim.freq, 49