Package 'OutSeekR'

January 20, 2025

Type Package

Title Statistical Approach to Outlier Detection in RNA-Seq and Related
Data

Version 1.0.0

Date 2024-11-15

Description An approach to outlier detection in RNA-seq and related data
based on five statistics. 'OutSeekR' implements an outlier test by comparing
the distributions of these statistics in observed data with those of
simulated null data.

Depends R (>= 2.10)

Imports future.apply, gamlss, gamlss.dist, lsa, truncnorm

Suggests future, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Jee Yun Han [aut],
John Sahrmann [aut],
Jaron Arbet [ctb],
Paul Boutros [aut, cre, cph]

Maintainer Paul Boutros <pboutros@mednet.ucla.edu>

Repository CRAN

Date/Publication 2024-11-19 09:10:05 UTC

Contents

---

calculate.p.values   *Calculate p-values*

Description

Calculate p-values for each sample of a single transcript.

Usage

```
calculate.p.values(
  x,
  x.distribution,
  x.zrange.mean,
  x.zrange.median,
  x.zrange.trimmean,
  x.fraction.kmeans,
  x.cosine.similarity,
  null.zrange.mean,
  null.zrange.median,
  null.zrange.trimmean,
  null.fraction.kmeans,
  null.cosine.similarity,
  kmeans.nstart = 1
)
```

Arguments

x	A numeric vector of values for an observed transcript.
x.distribution	A numeric code corresponding to the optimal distribution of x as returned by identify.bic.optimal.data.distribution().

x.zrange.mean   A number, the range of the z-scores calculated using the mean and standard deviation of x.

x.zrange.median

A number, the range of the z-scores calculated using the median and median absolute deviation of x.

x.zrange.trimmean

A number, the range of the z-scores calculated using the trimmed mean and trimmed standard deviation of x.

x.fraction.kmeans

A number, the k-means fraction of x.

x.cosine.similarity

A number, the cosine similarity of x.

null.zrange.mean

A numeric vector, the ranges of the z-scores calculated using the mean and standard deviation of each transcript in the null data.

null.zrange.median

A numeric vector, the ranges of the z-scores calculated using the median and median absolute deviation of each transcript in the null data.

null.zrange.trimmean

A numeric vector, the ranges of the z-scores calculated using the trimmed mean and trimmed standard deviation of each transcript in the null data.

null.fraction.kmeans

A numeric vector, the k-means fraction of each transcript in the null data.

null.cosine.similarity

A numeric vector, the cosine similarity of each transcript in the null data.

kmeans.nstart   The number of random starts when computing k-means fraction; default is 1. See ?stats::kmeans for further details.

Value

A list consisting of the following entries:

- p.values: a vector of p-values for the outlier test run on each sample (up until the p-value exceeds p.value.threshold); and

- outlier.statistics.list, a list of vectors containing the values of the outlier statistics calculated from the remaining samples. The list will be of length equal to one plus the total number of outliers (i.e., the number of samples with an outlier test p-value less than p.value.threshold) and will contain entries outlier.statistics.N, where N is between zero and the total number of outliers. outlier.statistics.N is the vector of outlier statistics after excluding the Nth outlier sample, with outlier.statistics.0 being for the complete transcript.

Examples

```
data(example.data.for.calculate.p.values);
i <- 1; # row index of transcript to test
calculate.p.values(
   x = example.data.for.calculate.p.values$data[i,],
```

```
x.distribution = example.data.for.calculate.p.values$x.distribution[i],
x.zrange.mean = example.data.for.calculate.p.values$x.zrange.mean[i],
x.zrange.median = example.data.for.calculate.p.values$x.zrange.median[i],
x.zrange.trimmean = example.data.for.calculate.p.values$x.zrange.trimmean[i],
x.fraction.kmeans = example.data.for.calculate.p.values$x.fraction.kmeans[i],
x.cosine.similarity = example.data.for.calculate.p.values$x.cosine.similarity[i],
null.zrange.mean = example.data.for.calculate.p.values$null.zrange.mean,
null.zrange.median = example.data.for.calculate.p.values$null.zrange.median,
null.zrange.trimmean = example.data.for.calculate.p.values$null.zrange.trimmean,
null.fraction.kmeans = example.data.for.calculate.p.values$null.fraction.kmeans,
null.cosine.similarity = example.data.for.calculate.p.values$null.cosine.similarity,
kmeans.nstart = example.data.for.calculate.p.values$kmeans.nstart
);
```

---

calculate.residuals   *Calculate residuals*

Description

Calculate residuals between quantiles of the input and quantiles of one of four distributions: normal, log-normal, exponential, or gamma.

Usage

```
calculate.residuals(x, distribution)
```

Arguments

x	A numeric vector.
distribution	A number corresponding to the optimal distribution of x as returned by, e.g., `identify.bic.optimal.data.distribution()`. One of

- 1 = normal,
- 2 = log-normal,
- 3 = exponential, and
- 4 = gamma.

Value

A numeric vector of the same length as x. Names are not retained.

Examples

```
# Generate fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
```

```
    );
names(x) <- paste(
    'Sample',
    seq_along(x),
    sep = '.'
    );
calculate.residuals(
    x = x,
    distribution = 4
    );
```

---

detect.outliers   *Detect outliers*

Description

Detect outliers in normalized RNA-seq data.

Usage

```
detect.outliers(
  data,
  num.null = 1000,
  initial.screen.method = c("fdr", "p.value"),
  p.value.threshold = 0.05,
  fdr.threshold = 0.01,
  kmeans.nstart = 1
)
```

Arguments

data	A matrix or data frame of normalized RNA-seq data, organized with transcripts on rows and samples on columns. Transcript identifiers should be stored as `rownames(data)`.
num.null	The number of transcripts to generate when simulating from null distributions; default is 1000. We recommend using at least 10,000 iterations for publication-level results, with 100,000 or even one million iterations providing more robust estimates.
initial.screen.method	
	The statistical criterion for initial gene selection; valid options are 'FDR' and 'p-value'.
p.value.threshold	
	The p-value threshold for the outlier test; default is 0.05. Once the p-value for a sample exceeds `p.value.threshold`, testing for that transcript ceases, and all remaining samples will have p-values equal to `NA`.
fdr.threshold	The false discovery rate (FDR)-adjusted p-value threshold for determining the final count of outliers; default is 0.01.

kmeans.nstart   The number of random starts when computing k-means fraction; default is 1.
                 See `?stats::kmeans` for further details.

Value

A list consisting of the following entries:

- `p.values`: a matrix of unadjusted p-values for the outlier test run on each transcript in `data`.
- `fdr`: a matrix of FDR-adjusted p-values for the outlier test run on each transcript in `data`.
- `num.outliers`: a vector giving the number of outliers detected for each transcript based on the threshold.
- `outlier.test.results.list`: a list of length `max(num.outliers) + 1` containing entries `roundN`, where N is between one and `max(num.outliers) + 1`. `roundN` is the data frame of results for the outlier test after excluding the (N-1)th outlier sample, with `round1` being for the original data set (i.e., before excluding any outlier samples).
- `distributions`: a numeric vector indicating the optimal distribution for each transcript. Possible values are 1 (normal), 2 (log-normal), 3 (exponential), and 4 (gamma).
- `initial.screen.method`: Specifies the statistical criterion for initial feature selection. Valid options are 'p-value' and 'FDR' (p-value used by default).

Examples

```
data(outliers);
outliers.subset <- outliers[1:10,];
results <- detect.outliers(
   data = outliers.subset,
   num.null = 10
   );
```

---

example.data.for.calculate.p.values
                    example.data.for.calculate.p.values

Description

Example data (list object) for testing `calculate.p.values()`.

Usage

```
example.data.for.calculate.p.values
```

Format

An object of class `list` of length 13.

---

identify.bic.optimal.data.distribution

Identify optimal distribution of data

Description

Identify which of four distributions—normal, log-normal, exponential, or gamma—best fits the given data according to BIC.

Usage

```
## S3 method for class 'bic.optimal.data.distribution'
identify(x)
```

Arguments

x   A numeric vector.

Value

A numeric code representing which distribution optimally fits x. Possible values are

- 1 = normal,

- 2 = log-normal,

- 3 = exponential, and

- 4 = gamma.

Examples

```
# Generate fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
    );
identify.bic.optimal.data.distribution(
    x = x
    );
```

---

identify.bic.optimal.residuals.distribution

Identify optimal distribution of residuals

Description

Identify which of four distributions—normal, log-normal, exponential, or gamma—best fits the given vector of residuals according to BIC.

Usage

```
## S3 method for class 'bic.optimal.residuals.distribution'
identify(x)
```

Arguments

x   A numeric vector.

Value

A numeric code representing which distribution optimally fits x. Possible values are

- 1 = normal,

- 2 = log-normal,

- 3 = exponential, and

- 4 = gamma.

Examples

```
# Generate fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
    );
identify.bic.optimal.residuals.distribution(
    x = x
    );
```

---

kmeans.fraction *k-means fraction*

Description

Given a vector of cluster assigments from `quantify.outliers()` run with `method = 'kmeans'`, compute the fraction of observations belonging to the smaller of the two clusters.

Usage

```
kmeans.fraction(x)
```

Arguments

x   A numeric vector.

Details

This function only considers clusters 1 and 2 even if `quantify.outliers()` was run with `exclude.zero = TRUE`. In that case, zeros are effectively excluded from the counts used to define the k-means fraction. See examples.

Value

A number.

Examples

```
x <- c(1, 1, 2, 2, 2, 2, 2, 2, 2, 2);
names(x) <- letters[1:length(x)];
kmeans.fraction(x);
```

---

outlier.detection.cosine
Cosine similarity

Description

Compute cosine similarity for detection of outliers. Generate theoretical quantiles based on the optimal distribution of the data, and compute cosine similarity between a point made up of the largest observed quantile and the largest theoretical quantile and a point on the line y = x. .

Usage

```
outlier.detection.cosine(x, distribution)
```

Arguments

x	A numeric vector.
distribution	A numeric code corresponding to the optimal distribution of x as returned by identify.bic.optimal.data.distribution().

Value

A number.

Examples

```
# Generate fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
    );
outlier.detection.cosine(
    x = x,
    distribution = 4
    );
```

---

outliers   *Example data set for outlier testing*

Description

Example data set for outlier testing

Usage

```
outliers
```

Format

A data frame with 500 rows and 50 columns:

S01 simulated fragments per kilobase of transcript per million fragments mapped (FPKM) values for sample 1

S02 simulated FPKM values for sample 2

S03 simulated FPKM values for sample 3

S04 simulated FPKM values for sample 4

S05 simulated FPKM values for sample 5

S06 simulated FPKM values for sample 6

S07 simulated FPKM values for sample 7

S08 simulated FPKM values for sample 8

S09 simulated FPKM values for sample 9

S10 simulated FPKM values for sample 10

S11 simulated FPKM values for sample 11

S12 simulated FPKM values for sample 12

S13 simulated FPKM values for sample 13

S14 simulated FPKM values for sample 14

S15 simulated FPKM values for sample 15

S16 simulated FPKM values for sample 16

S17 simulated FPKM values for sample 17

S18 simulated FPKM values for sample 18

S19 simulated FPKM values for sample 19

S20 simulated FPKM values for sample 20

S21 simulated FPKM values for sample 21

S22 simulated FPKM values for sample 22

S23 simulated FPKM values for sample 23

S24 simulated FPKM values for sample 24

S25 simulated FPKM values for sample 25

S26 simulated FPKM values for sample 26

S27 simulated FPKM values for sample 27

S28 simulated FPKM values for sample 28

S29 simulated FPKM values for sample 29

S30 simulated FPKM values for sample 30

S31 simulated FPKM values for sample 31

S32 simulated FPKM values for sample 32

S33 simulated FPKM values for sample 33

S34 simulated FPKM values for sample 34

S35 simulated FPKM values for sample 35

S36 simulated FPKM values for sample 36

S37 simulated FPKM values for sample 37

S38 simulated FPKM values for sample 38

S39 simulated FPKM values for sample 39

S40 simulated FPKM values for sample 40

S41 simulated FPKM values for sample 41

S42 simulated FPKM values for sample 42

S43 simulated FPKM values for sample 43

S44 simulated FPKM values for sample 44

S45 simulated FPKM values for sample 45

S46 simulated FPKM values for sample 46

S47 simulated FPKM values for sample 47

S48 simulated FPKM values for sample 48

S49 simulated FPKM values for sample 49

S50 simulated FPKM values for sample 50

---

quantify.outliers   *Compute quantities for outlier detection*

Description

Compute quantities for use in the detection of outliers. Specifically, compute z-scores based on the mean / standard deviation, the trimmed mean / trimmed standard deviation, or the median / median absolute deviation, or the cluster assignment from k-means with two clusters.

Usage

```
quantify.outliers(
  x,
  method = "mean",
  trim = 0,
  nstart = 1,
  exclude.zero = FALSE
)
```

Arguments

x	A numeric vector.
method	A string indicating the quantities to be computed. Possible values are

- `'mean'` : z-scores based on mean and standard deviation or trimmed mean and trimmed standard deviation if `trim > 0`,
- `'median'` : z-scores based on median and median absolute deviation, or
- `'kmeans'` : cluster assignment from k-means with two clusters. The default is z-scores based on the mean and standard deviation.

trim	A number, the fraction of observations to be trimmed from each end of x. Default is no trimming.
nstart	A number, for k-means clustering, the number of random initial centers for the clusters. Default is 1. See stats::kmeans() for further information.
exclude.zero	A logical, whether zeros should be excluded (`TRUE`) or not excluded (`FALSE`, the default) from computations. For `method = 'mean'` and `method = 'median'`, this means zeros will not be included in computing the summary statistics; for `method = 'kmeans'`, this means zeros will be placed in their own cluster, coded 0.

Value

A numeric vector the same size as x whose values are the requested quantities computed on the corresponding elements of x.

Examples

```
# Generate fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
    );
# Add missing values and zeros for demonstration.  Missing values are
# ignored, and zeros can be ignored with `exclude.zeros = TRUE`.
x[1:5] <- NA;
x[6:10] <- 0;

# Compute z-scores based on mean and standard deviation.
quantify.outliers(
    x = x,
    method = 'mean',
    trim = 0
    );
# Exclude zeros from the calculation of the mean and standard
# deviation.
quantify.outliers(
    x = x,
    method = 'mean',
    trim = 0,
    exclude.zero = TRUE
    );

# Compute z-scores based on the 5% trimmed mean and 5% trimmed
# standard deviation.
quantify.outliers(
    x = x,
    method = 'mean',
    trim = 0.05
    );

# Compute z-scores based on the median and median absolute deviation.
quantify.outliers(
    x = x,
    method = 'median'
    );

# Compute cluster assignments using k-means with k = 2.
quantify.outliers(
    x = x,
    method = 'kmeans'
    );
```

```
# Try different initial cluster assignments.
quantify.outliers(
    x = x,
    method = 'kmeans',
    nstart = 10
    );
# Assign zeros to their own cluster.
quantify.outliers(
    x = x,
    method = 'kmeans',
    exclude.zero = TRUE
    );
```

---

simulate.null   *Simulate from a null distribution*

Description

Simulate transcripts from a specified null distribution.

Usage

```
## S3 method for class 'null'
simulate(x, x.distribution, r, r.distribution)
```

Arguments

x   A numeric vector of transcripts.

x.distribution   A numeric code corresponding to the optimal distribution of x as returned by
                  identify.bic.optimal.data.distribution(). Possible values are

                  • 1 = normal,
                  • 2 = log-normal,
                  • 3 = exponential, and
                  • 4 = gamma.

r   A numeric vector of residuals calculated for this transcript.

r.distribution   A numeric code corresponding to the optimal distribution of x as returned by
                  identify.bic.optimal.residuals.distribution(). Possible values are the
                  same as those for x.distribution.

Value

A numeric vector of the same length as x. Names are not retained.

Examples

```
# Prepare fake data.
set.seed(1234);
x <- rgamma(
    n = 20,
    shape = 2,
    scale = 2
    );
names(x) <- paste('Sample', seq_along(x), sep = '.');
x.dist <- identify.bic.optimal.data.distribution(
    x = x
    );
r <- calculate.residuals(
    x = x,
    distribution = x.dist
    );
r.trimmed <- trim.sample(
    x = r
    );
r.dist <- identify.bic.optimal.residuals.distribution(
    x = r.trimmed
    );
null <- simulate.null(
    x = x,
    x.distribution = x.dist,
    r = r.trimmed,
    r.distribution = r.dist
    );
```

---

trim.sample	*Trim a vector of numbers*

Description

Symmetrically trim a vector of numbers after sorting it.

Usage

```
trim.sample(x, trim = 0.05)
```

Arguments

x	A numeric vector.
trim	A number, the fraction of observations to be trimmed from each end of x.

Details

If length(x) <= 10, the function returns x[2:(length(x) - 1)].

Value

A sorted, trimmed copy of x.

Examples

```
trim.sample(
    x = 1:20,
    trim = 0.05
    );
```

---

zrange   *Range of z-scores*

Description

Compute the range of a vector of z-scores.

Usage

```
zrange(x)
```

Arguments

x   A numeric vector

Value

A number.

Examples

```
set.seed(1234);
x <- rnorm(
    n = 10
    );
zrange(
    x = x
    );
```

Index