Package ‘rdi’

July 23, 2025

Type Package

Version 1.0.0

Date 2018-05-01

Title Repertoire Dissimilarity Index

Description Methods for calculation and visualization of the Repertoire Dissimilarity Index. Citation: Bolen and Rubelt, et al (2017) <doi:10.1186/s12859-017-1556-5>.

License CC BY-SA 4.0

URL `http://rdi.readthedocs.io`

BugReports `https://bitbucket.org/cbolen1/rdicore/issues`

LazyData true

Depends R (>= 3.0.0)

Imports beanplot, gplots, pdist, stringr

Suggests knitr, ggplot2

RoxygenNote 6.0.1

NeedsCompilation no

Author Christopher Bolen [aut, cre],
Florian Rubelt [aut],
Jason Vander Heiden [aut]

Maintainer Christopher Bolen <`cbolen1@gmail.com`>

Repository CRAN

Date/Publication 2018-05-07 11:14:21 UTC

Contents

calcRDI *Calculate repertoire distances*

Description

Calculate repertoire distances from a matrix of vdjCounts

Usage

```
calcRDI(vdjCounts, distMethod = c("euclidean", "cor"), subsample = TRUE,
  nIter = 100, constScale = TRUE, units = c("lfc", "pct"), ...)
```

Arguments

`vdjCounts`	a matrix of repertoire counts, as created by calcVDJCounts
`distMethod`	one of c("euclidean","cor") determining how to calculate the distance from the matrix of vdj counts. See Details.
`subsample`	logical; if true, all repertoires are subsampled to be equal size.
`nIter`	value defining how many times the subsampling should be repeated. Only used if subsample is TRUE.
`constScale`	logical; if `TRUE`, vdjCounts will be scaled such that the sum of each column will be equal to 500 counts (an arbitrary constant). Otherwise, the columns will be scaled to the average count of all the columns.
`units`	One of "lfc" or "pct". This determines the method used for transforming the repertoire counts. See Details.
`...`	additional parameters; these are ignored by this function.

Details

There are two options for distance methods, "euclidean" and "cor". Euclidean refers to standard euclidean distance, and is the standard for the RDI measure described in (Bolen et al. Bioinformatics 2016). In contrast, cor refers to a correlation-based distance metric, where the distance is defined as `(1-correlation)` between each column of vdjCounts.

The `units` parameter is used to determine the transformation function for the repertoire counts. If `units='lfc'` (default), then the arcsinh transformation is applied to the count matrix, resulting in a distance metric which will scale with the average log fold change of each gene. In contrast, `units='pct'` will result in no transformation of the count matrix, and distances will be proportional to the average percent change of each gene, instead. Note that "units" is a bit of a misnomer, as the distance metric doesn't actually represent the true log-fold or percent change in the repertoires. In order to actually estimate these parameters, refer to the rdiModel and convertRDI functions.

Value

A dissimilarity structure containing distances between repertoires, averaged across each subsampe run. In addition to the standard attributes in a dist object, three additional attributes are defined as follows:

ngenes	integers, the number of genes in each column of "genes" that were included in at least one repertoire.
nseq	integer, the number of sequences used after subsampling the repertoires. If `subsample=FALSE`, this is not defined.
units	string, either "lfc" or "pct", depending on the "units" in the original call

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)

#create sequence annotations
seqAnnot = data.frame(donor = sample(1:4, 10000, replace=TRUE),
                      cellType = sample(c("B","T"), 10000, replace=TRUE)
                     )
##generate repertoire counts
cts = calcVDJcounts(genes,seqAnnot)

##calculate RDI
d = calcRDI(cts)

##calculate RDI in percent space
d_pct = calcRDI(cts,units="pct")

##convert RDI to actual 'lfc' estimates and compare
dtrue = convertRDI(d)$pred
plot(d, dtrue)
```

`calcVDJcounts` *Calculate repertoire counts*

Description

Create count matrices for a set of repertoires

Usage

```
calcVDJcounts(genes, seqAnnot, select = NULL, combine = NULL,
  vdjDrop = NULL, splitBy = NULL, simplifyNames = TRUE,
  splitCommas = FALSE, ...)
```

Arguments

`genes`	vector (or matrix) containing the gene calls for each sequence. If genes is a matrix, counts will be calculated for each column of 'genes', and the resulting count matrices will be concatenated. See Details.
`seqAnnot`	matrix containing repertoire annotations.
`select`	a list containing definitions of which repertoires to use. See Details.
`combine`	a list defining repertoires that should be grouped together. See Details.
`vdjDrop`	a list specifying specific genes to exclude from analysis. See Details.
`splitBy`	the columns in seqAnnot to use for splitting repertoires. Default is to use all columns in seqAnnot.
`simplifyNames`	logical; if true, any columns of seqAnnot where all selected (and collapsed) sequences share the same value will not be used to make the names of sequenceCodes.
`splitCommas`	logical; if true, seqAnnot is assumed to contain. comma-separated lists of sequence annotations, which will be split up before generating sequence codes. Note: setting this to TRUE will make processing much slower.
`...`	additional parameters; these are ignored by this function.

Details

In most cases, `genes` will be a single vector or one-column matrix. However, there are some cases where a row of `seqAnnot` corresponds to two (or more) genes (e.g. the V and J gene segments of a single immune sequence). Rather than make multiple rows for each gene, the `calcVDJcounts` function provides the option to provide a multi-column matrix for `genes`. The counts for each column will be tallied separately, and are then concatenated.

To ensure equal variance across all repertoires, the default RDI metric uses subsampling to ensure that all repertoires have the same number of sequences. The default RDI metric subsamples all repertoires to the size of the smallest repertoire, which may result in a loss of power for comparisons between larger repertoires. In order to increase power for various tests, it is often useful to only calculate the repertoire counts for a subset of the repertoires in seqAnnot. This can be done by using the `select` and `combine` parameters to specify which repertoires to include in the analysis.

Both parameters are lists containing entries with the same name as one of the columns of seqAnnot. For `select`, each entry is a vector defining which values to include (e.g., to include only Visit 1 and 3, you might specify `select=list(visit=c("V1","V3"))`, where the `'visit'` column in seqAnnot contains the values `"V1"`,`"V2"`, and `"V3"`). In this case, any rows of `genes` and `seqAnnot` that come from a repertoire not specified in `select` will be discarded. By default, if a `select` code is not specified for a column in `seqAnnot`, all values from that column will be included.

The `combine` parameter works in a similar fashion, but instead of a vector describing which parameters to include, you can specify a vector of regular expressions, and any values of the `seqAnnot` column that match the regular expression will be combined into a single repertoire (e.g. to combine visits 1 and 3 into a single repertoire, you might specify `combine=list(visit="V[13]")`).

The `vdjDrop` parameter is also useful for limiting sequences. Like `select` and `combine`, this is a named list, with entries corresponding to the columns of `genes`. Each entry of `vdjDrop` is a vector of gene segment names to remove from the analysis. All sequences containing those genes are removed from the analysis before subsampling.

Once unwanted rows have been removed, the columns of `seqAnnot` are concatenated to generate "repertoire" labels for each row. The repertoire labels are then used to split the rows of `genes`, and gene prevalence is tallied within a repertoire. By default, columns of `seqAnnot` that are constant after subsetting will not be included in the label. However, this can be controlled by the `simplifyNames` parameter. If `simplifyNames` is FALSE, all columns of `seqAnnot` are included when generating labels.

Value

A matrix where each row represents a gene, and each column represents a repertoire.

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)

#create sequence annotations
seqAnnot = data.frame(donor = sample(1:4, 10000, replace=TRUE),
                      visit = sample(c("V1","V2","V3"), 10000, replace=TRUE),
                      cellType = sample(c("B","T"), 10000, replace=TRUE)
                     )

##generate repertoire counts for all repertoires
cts = calcVDJcounts(genes,seqAnnot)

##Only include visit 1
cts = calcVDJcounts(genes,seqAnnot, select=list(visit="V1"))


## Just T cell repertoires, combining visit 1 and 3 together, and dropping visit 2
cts = calcVDJcounts(genes,seqAnnot,
                    select=list(cellType="T", visit=c("V1","V3")),
                    combine=list(visit="V[13]"))
```

`convertRDI` *Convert RDI measures*

Description

Method to convert RDI values to fold/percent change

Usage

```
convertRDI(d, models = NULL, calcSD = FALSE)
```

Arguments

d	Distance matrix (as produced by calcRDI), or a vector of distances.
models	Set of RDI models, as produced by rdiModel. If `NULL`, RDI models will be calculated based on the attributes in the distance matrix.
calcSD	logical; if `TRUE`, standard deviations for each estimate will be returned.

Details

The convertRDI function works by first generating a model for the RDI values at a given repertoire size and feature count using the rdiModel function (see that method's help file for more details). The RDI models predict the average log-fold/percent change across a range of RDI values, and allows us to convert RDI to a more stable and interpretable metric.

In addition to the average log-fold or percent change value, rdiModel also generates models for the standard deviation at each RDI value. This is useful for understanding the confidence intervals around the fold change estimate.

Value

A list containing either one or two features:

pred	The converted predictions; same length as `d`.
sd	If `calcSD==T`, a set of standard deviation estimates for each prediction.

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)
#create sequence annotations
seqAnnot = data.frame(donor = sample(1:4, 10000, replace=TRUE))
#calculate RDI
d = rdi(genes, seqAnnot)

##convert RDI to actual 'lfc' estimates and compare
dtrue = convertRDI(d)$pred
plot(d, dtrue)

##look at SD ranges around lfc estimates
dtrue = convertRDI(d, calcSD=TRUE)

##plot using ggplot2
library(ggplot2)
x = as.numeric(d)
y = as.numeric(dtrue$pred)
sd = as.numeric(dtrue$sd)
qplot(x,y)+geom_errorbar(aes(x=x, ymin=y-sd, ymax=y+sd))
```

`plotRDIladder` *RDI ladder plotting function*

Description

function for adding a pre-computed RDI ladder onto a plot

Usage

```
plotRDIladder(ladder, side = 4, toPlot = NULL, labelLadder = TRUE,
  add = TRUE, cex = 0.7, lineCol = NULL, fillCol = "#AAAAAA")
```

Arguments

`ladder`	the ladder object to add, as created by rdiLadder
`side`	integer; value between 1 and 4 indicating where the ladder will be added. 1 - bottom, 2 - left, 3 - top, 4 - right.
`toPlot`	logical vector; which ladders should be plotted? By default, ladders that are significantly overlapped by their neighbor and those that are majority outside the plotting region are removed.
`labelLadder`	logical; if `TRUE`, each curve will be annotated with the ladder name
`add`	logical; if `TRUE`, the ladder will be added to the current plot
`cex`	character expansion for ladder labels.
`lineCol`	the colors to be used for the ladder border. If the length of `col` is less than the length of ladder, `col` will be recycled.
`fillCol`	the colors to be used to fill the ladder. If the length of `col` is less than the length of ladder, `col` will be recycled.

Details

This function is used in conjunction with rdiLadder to add a useful annotation to any plot containing RDI values.

Because RDI values vary according to the number of genes and size of the repertoires, they are not useful as numbers by themselves. Instead, it is useful to compare them with estimates of the true difference between the two repertoires. This function adds a series of density curves along one side of a standard plotting region, each one representing the most likely RDI values between two repertoires that vary by a set amount.

By default, not all density curves from the `ladder` parameter are plotted. Instead, the function intelligently chooses which ladders to plot based on the amount of overlap between neighboring ladders. If a ladder is significantly overlapped by the ladder below it, then the ladder will not be plotted. In addition, if the mean of a ladder is outside the main plotting region, it will be dropped. In order to control this behavior, you can directly specify which ladders are plotted using the `toPlot` parameter.

Value

Invisibly returns the location of the ladder (if side 1 or 3, the y location; otherwise, the x location).

See Also

rdiLadder, rdiModel, rdiAxis

rdi — *Calculate RDI dissimilarity matrix*

Description

Wrapper function for calculating RDIs

Usage

```
rdi(genes, seqAnnot, params = NULL, ...)
```

Arguments

`genes`	vector (or matrix) containing the gene calls for each sequence. If genes is a matrix, counts will be calculated for each column of 'genes', and the resulting count matrices will be concatenated.
`seqAnnot`	matrix containing repertoire annotations. Must be same length as 'genes'.
`params`	list; contains parameters to pass to child functions. Should contain `countParams` and `distParams` lists, which contain parameters for calcVDJcounts and calcRDI, respectively. See Details.
`...`	other parameters to pass to calcVDJcounts and calcRDI.

Details

This function is a wrapper for the two core functions of RDI, calcVDJcounts and calcRDI. To control the function of both `calcVDJcounts` and `calcRDI`, additional parameters can be specified either directly in the RDI function call, or parameters for the individual functions can be wrapped up into lists of parameters and passed into the `params` parameter. `params` should be a list containing at least one of two parameter lists: `countParams` and `distParams`, which will be passed to `calcVDJcounts` and `calcRDI`, respectively. An example analysis is included below.

Value

Dissimilarity structure, as calculated by dist. In addition to the standard attributes returned by dist, two additional attributes are defined as follows:

nseq	integer, the number of sequences used after subsampling the repertoires
ngenes	integers, the number of genes in each column of "genes" that were included in at least one repertoire.

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)

#create sequence annotations
seqAnnot = data.frame(donor = sample(1:4, 10000, replace=TRUE),
                      visit = sample(c("V1","V2","V3"), 10000, replace=TRUE),
                      cellType = sample(c("B","T"), 10000, replace=TRUE)
                     )

#parameters
params = list(
  countParams = list(
    select = list(
      visit = c("V1","V3"),
      cellType = "B"
    ),
    combine = list(
      visit = "V[13]"
    ),
    simplifyNames = FALSE
  ),
  distParams = list(
    constScale=FALSE
  )
)

##calculate RDI
d = rdi(genes, seqAnnot, params)

##plot using hierarchical clustering
plot(hclust(d))
```

`rdiAxis` *RDI Axis annotation function*

Description

This function takes a RDI model, as generated by rdiModel, and adds an axis with annotations in the fold change space.

Usage

```
rdiAxis(model, side = 2, at = NULL, ...)
```

Arguments

`model`	A model object, as generated by rdiModel.

`side`	The side the axis will be added to. (1 - bottom; 2 - left; 3 - top; 4 - right). Default is 2.
`at`	The points at which the tick marks are drawn. By default, tickmarks are placed at 'round' fold/percent change values using the "pretty" breakpoints function. This may not be ideal if log-RDI values are being plotted.
`...`	Additional parameters to pass to axis

Details

This function is designed to replace the default axes generated by a plot function. Instead of annotating the true RDI value, `rdiAxis` will estimate the "true difference" values at various points within the plotting region, and will annotate the axis with those estimates.

It is worth noting that although the RDI value can range below `rdiModel`'s estimate for "identical" repertoires, no negative true difference values will be annotated, as these values do not make sense.

See Also

rdiModel, rdiLadder, plotRDIladder

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)
#create sequence annotations
seqAnnot = data.frame(donor = sample(1:10, 10000, replace=TRUE))
#calculate RDI
d = rdi(genes, seqAnnot)

##create a "baseVect" with the same probability as our features
##since we sampled uniformly, the base vector has equal probability
baseVect = rep(1/length(letters),length(letters))

##generate an RDI model
m = rdiModel(attr(d, "nseq"), baseVects=baseVect)

##convert RDI to lfc
td = convertRDI(d,models=m)$pred

par(mar=c(4,4,1,4),las=1,mgp=c(3,0.5,0))
plot(td,d, ylab="RDI", xlab="LFC")

##now add "true difference" axis annotation to the right side of the plot
rdiAxis(m, side=4)
```

`rdiLadder` *RDI ladder*

Description

Function for creating the RDI ladder for a specific number of sequences

Usage

```
rdiLadder(n, ngenes = NULL, baseVects = NULL, diffPoints = NULL,
  units = c("lfc", "pct"), ...)
```

Arguments

`n`	the repertoire size; alternatively, an rdiModel object as created by rdiModel.
`ngenes`	numeric vector indicating the number of genes in each chain. If baseVect is not provided, this parameter will be used to generate a base prevalence vector for each of the genes.
`baseVects`	A vector or list of vectors representing the total prevalence of each gene (for each chain) in the dataset. See rdiModel for details.
`diffPoints`	numeric vector; each value specifices either a log2 fold change or percent deviation value (depending on the 'units') at which the RDI ladder will be calculated.
`units`	String; either "lfc" or "pct", depending on what transform was used in the original RDI calculations. See Details.
`...`	Additional parameters to be passed to rdiModel

Details

Because RDI values vary according to the number of genes and size of the repertoires, they are not useful as numbers by themselves. Instead, it is useful to compare them with estimates of the true difference between the two repertoires. This function uses the models generated by rdiModel to generate estimated RDI values corresponding to a set of pre-defined true distance (log-fold change or percent) values. This function is primarily meant to be used in conjunction with plotRDIladder in order to add a useful reference point for RDI values.

The units used for the RDI model should always match the units used to generate your RDI values. For more details on units, refer to the details of calcRDI

Value

A list of the same length as diffPoints, with each entry in the list containing the mean RDI value and standard deviation corresponding to a given true difference value.

See Also

plotRDIladder, rdiModel, rdiAxis

`rdiModel` *RDI Models*

Description

Generate models equating RDI values to true differences in underlying prevalence values

Usage

```
rdiModel(n, ngenes = NULL, baseVects = NULL, nIter = 50, nSample = 20,
  units = c("lfc", "pct"), constScale = TRUE)
```

Arguments

`n`	the repertoire size.
`ngenes`	numeric vector indicating the number of genes in each chain. If `baseVects` is not provided, this parameter will be used to generate a base prevalence vector for each of the genes.
`baseVects`	A vector or list of vectors representing the total prevalence of each gene (for each chain) in the dataset. Differential datasets will be created from alterations of this vector. If not provided, a base vector will be randomly . generated at each subsample step containing the number of genes specified by ngenes.
`nIter`	The number of iterations (i.e. number of datasets to generate).
`nSample`	The number of samples to generate for each subsample. Each sample will have a different true fold change, but the same starting vector
`units`	String; either "lfc" or "pct", depending on what transform was used in the original RDI calculations. See Details.
`constScale`	logical; if `TRUE`, vdjCounts will be scaled such that the sum of each column will be equal to 500 (an arbitrary constant). Otherwise, the columns will be scaled to the average count of all the columns.

Details

This method uses simulated sequencing datasets to estimate the RDI values for datasets with a known true deviation.

Briefly, a baseline probability vector (either randomly generated or supplied by the `baseVects` parameter) is randomly perturbed, and the difference between the baseline vector and the perturbed vector is calculated. Then, `nSample` sequencing datasets of size n are randomly drawn from both the baseline vector and the perturbed vector, and the RDI distance between all datasets calculated. This process is repeated `nIter` times, resulting in a dataset of RDI values and matched true differences. A set of spline models is then fit to the data: one from RDI to true difference, and another from true difference to RDI value, allowing for bi-directional conversions.

If a baseline probability vector is not provided, one will be generated from an empirical model of gene segment prevalence. However, for best performance, this is not recommended. Estimates of true fold change is very sensitive to the distribution of features in your count dataset, and it is important that your baseline vector match your overall dataset as accurately as possible. The best baseline vector is almost always the average feature prevalence across all repertoires in a dataset, although manually generated baseline vectors may also work well.

The units used for the RDI model should always match the units used to generate your RDI values. For more details on units, refer to the details of calcRDI.

Value

A list containing three objects:

fit	an object of class `"smooth.spline"`, based on a spline model with the true difference (lfc or pct) as the independent (
rev.fit	an object of class `"smooth.spline"`. The opposite of fit. Used for converting from RDI to true difference.
units	one of `c("lfc","pct")`, representing the units of the true difference values.

See Also

rdiAxis, rdiLadder, plotRDIladder

Examples

```
#create genes
genes = sample(letters, 10000, replace=TRUE)
#create sequence annotations
seqAnnot = data.frame(donor = sample(1:4, 10000, replace=TRUE))
#calculate RDI
d = rdi(genes, seqAnnot)

##create a "baseVect" with the same probability as our features
##since we sampled uniformly, the base vector has equal probability
baseVect = rep(1/length(letters),length(letters))

##generate an RDI model
m = rdiModel(attr(d, "nseq"), baseVects=baseVect)

##plot the spline model
plot(m$fit, xlab="log fold change",ylab="RDI",type='l')

##convert RDI to log fold change
mean = predict(m$rev.fit, d)$y
mean[mean<0] = 0
```

Index