Package ‘powerTCR’

January 30, 2026

Type Package

Title Model-Based Comparative Analysis of the TCR Repertoire

Version 1.30.0

Date 2018-01-23

Author Hillary Koch

Maintainer Hillary Koch <hillary.koch01@gmail.com>

Description This package provides a model for the clone size distribution of the TCR repertoire. Further, it permits comparative analysis of TCR repertoire libraries based on theoretical model fits.

License Artistic-2.0

Encoding UTF-8

LazyData true

Imports cubature, doParallel, evmix, foreach, magrittr, methods, parallel, purrr, stats, truncdist, vegan, VGAM

Suggests BiocStyle, knitr, rmarkdown, RUnit, BiocGenerics

VignetteBuilder knitr

biocViews Software, Clustering, BiomedicalInformatics

git_url https://git.bioconductor.org/packages/powerTCR

git_branch RELEASE_3_22

git_last_commit d26ba80

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-01-29

Contents

`Accessors` *Grab important output from a list of fits*

Description

These are convenient accessors that will grab important output from a list of fits from fdiscgammagpd. They will grab the maximum likelihood estimates and/or the negative log likelihood for the maximum likelihood estimates.

Usage

```
get_mle(fits)
get_nllh(fits)
get_diversity(fits)
```

Arguments

`fits`	A list of fits output from fdiscgammagpd.

Value

A list of out either maximum likelihood estimates (get_mle) or negative log likelihoods (get nllh) corresponding to the list of fits. For get_diversity, a data frame of diversity estimates (species richness, Shannon entropy, clonality, and proportion of highly stimulated clones) for the samples.

Author(s)

`<hbk5086@psu.edu>`

See Also

`fdiscgammagpd`

Examples

```
# Here is a good workflow using fdiscgammagpd:
# Choose quantiles for every sample repertoire in the same manner.
# Then fit the model in the same manner as well.

data("repertoires")

thresholds <- list()
fits <- list()
for(i in 1:2){
    thresholds[[i]] <- unique(round(quantile(repertoires[[i]], c(.8,.85,.9,.95))))
    fits[[i]] <- fdiscgammagpd(repertoires[[i]], useq = thresholds[[i]],
                               shift = min(repertoires[[i]]))
}
names(fits) <- c("fit1", "fit2")

mles <- get_mle(fits)
nllhs <- get_nllh(fits)
diversity_ests <- get_diversity(fits)

mles
nllhs
diversity_ests
```

`clusterPlot` *Visualize hierarchical clustering of samples*

Description

This function is just a simple wrapper for the hclust function. It takes a symmetrix matrix displaying pairwise distances between samples and outputs a plot of the hierarchical clustering using specified linkage. Note that the distances must be given as a matrix object, not a distance object.

Usage

```
clusterPlot(distances, method = c("complete", "ward.D", "ward.D2", "single",
             "average", "mcquitty", "median", "centroid"))
```

Arguments

`distances`	A symmetric matrix containined the Jensen-Shannon distance between pairs of distributions.
`method`	Linkage method, as in hclust

Value

A basic plot of the induced hierarchical clustering.

Note

The distances must be given as a matrix object, not a distance object. The distance between a distribution and itself is 0. This corresponds to a matrix diagonal of 0.

Author(s)

`<hbk5086@psu.edu>`

See Also

JS_spliced, JS_desponds

Examples

```
# Simulate 3 sampled individuals
set.seed(123)
s1 <- rdiscgammagpd(1000, shape = 3, rate = .15, u = 25, sigma = 15, xi = .5, shift = 1)
s2 <- rdiscgammagpd(1000, shape = 3.1, rate = .14, u = 26, sigma = 15, xi = .6, shift = 1)
s3 <- rdiscgammagpd(1000, shape = 10, rate = .3, u = 45, sigma = 20, xi = .7, shift = 1)

# Fit model to the data at the true thresholds
fits <- list("fit1" = fdiscgammagpd(s1, useq = 25),
             "fit2" = fdiscgammagpd(s2, useq = 26),
             "fit3" = fdiscgammagpd(s3, useq = 45))

# Compute the pairwise JS distance between 3 fitted models
distances <- matrix(rep(0, 9), nrow = 3)
colnames(distances) <- rownames(distances) <- c("s1", "s2","s3")
grid <- min(c(s1,s2,s3)):10000
for(i in 1:2){
    for(j in (i+1):3){
        distances[i,j] <- JS_spliced(grid,
                                     shiftp = min(fits[[i]]$x),
                                     shiftq = min(fits[[j]]$x),
                                     phip = fits[[i]]$mle['phi'],
                                     phiq = fits[[j]]$mle['phi'],
                                     shapep = fits[[i]]$mle['shape'],
                                     shapeq = fits[[j]]$mle['shape'],
                                     ratep = fits[[i]]$mle['rate'],
                                     rateq = fits[[j]]$mle['rate'],
                                     threshp = fits[[i]]$mle['thresh'],
                                     threshq = fits[[j]]$mle['thresh'],
                                     sigmap = fits[[i]]$mle['sigma'],
                                     sigmaq = fits[[j]]$mle['sigma'],
                                     xip = fits[[i]]$mle['xi'],
                                     xiq = fits[[j]]$mle['xi'])
    }
}

# Distances are symmetric
distances <- distances + t(distances)

# Perform clustering. Note that s1 and s2 were generated using similar
# parameters, so we might expect them to be clustered together
## Not run: clusterPlot(distances, method = c("ward.D"))
```

`discgammagpd` *The discrete gamma-GPD spliced threshold distribution*

Description

Density, distribution function, quantile function and random generation for the discrete gamma-GPD spliced threshold distribution. The distribution has gamma bulk with shape equal to `shape` and rate equal to `rate`. It is spliced at a threshold equal to `u` and has a GPD tail with sigma equal to `sigma` and xi equal to `xi`. The proportion of data above the threshold phi is equal to `phiu` and the data are shifted according to `shift`.

Usage

```
ddiscgammagpd(x, fit, shape, rate, u, sigma, xi, phiu = NULL, shift = 0, log = FALSE)
pdiscgammagpd(q, fit, shape, rate, u, sigma, xi, phiu = NULL, shift = 0)
qdiscgammagpd(p, fit, shape, rate, u, sigma, xi, phiu = NULL, shift = 0)
rdiscgammagpd(n, fit, shape, rate, u, sigma, xi, phiu = NULL, shift = 0)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`fit`	A fit output from fdiscgammagpd. If this object is passed, all parameter fields will automatically populate with the maximum likelihood estimates for the parameters in fit.
`shape`	shape parameter alpha of the truncated gamma distribution.
`rate`	rate parameter beta of the gamma distribution.
`u`	threshold.
`sigma`	scale parameter sigma of the GPD.
`xi`	shape parameter xi of the GPD
`phiu`	Propotion of data greater than or equal to threshold u.
`shift`	value the complete distribution is shifted by. Ideally, this is the smallest value of the count data from one sample.
`log`	Logical; if TRUE, probabilities p are given as log(p).

Details

The shape, rate, u, sigma, and xi parameters must be specified by the user. If `phiu` is left unspecified, it defaults to 1 minus the distribution function of a discrete gamma distribution (not a discrete truncated gamma) evaluated at `u-1`.

Value

`ddiscgammagpd` gives the density, `pdiscgammagpd` gives the distribution function, `qdiscgammagpd` gives the quantile function, and `rdiscgammagpd` generates random variables from the described distribution.

Author(s)

`<hbk5086@psu.edu>`

Examples

```
# Generate and sort a random sample for a log-log plot
d <- rdiscgammagpd(100, shape = 5, rate = .25, u = 25,
                    sigma = 15, xi = .5, shift = 1)
d <- sort(d, decreasing = TRUE)
plot(log(d), log(1:100))

# When phiu is specified to .2, exactly 80%
# of the data are below the threshold u
```

```
pdiscgammagpd(24, shape = 5, rate = .25, u = 25,
                sigma = 15, xi = .5, phiu = .2, shift = 1)

# Plot simulated data versus theoretical quantiles
quantiles <- qdiscgammagpd((100:1)/101, shape = 5, rate = .25, u = 25,
                            sigma = 15, xi = .5, shift = 1)
plot(log(d), log(quantiles))
abline(0,1) # The line x=y

# Density below shift value should be 0
ddiscgammagpd(0, shape = 5, rate = .25, u = 25, sigma = 15, xi = .5, shift = 1)

# This is an example of using the "fit" input instead of manually specifying all parameters
data("repertoires")
thresholds1 <- unique(round(quantile(repertoires[[1]], c(.75,.8,.85,.9,.95))))
fit1 <- fdiscgammagpd(repertoires[[1]], useq = thresholds1, shift = min(repertoires[[1]]))
qdiscgammagpd(.6, fit1)
```

`fdesponds` *Fit the type-I Pareto distribution as according to Desponds et al. (2016)*

Description

This function fits a continuous type-I pareto distribution to a vector of count data. Given data x, a threshold Cmin, and letting n be the number of clones greater than u, the shape parameter alpha is computed as

$n * 1/(\sum log(x/Cmin)) + 1.$

The method considers every possible threshold (that is, every element of the vector unique(x)). The threshold and alpha which minimize the Kolmogorov-Smirnov statistic are selected.

Usage

```
fdesponds(x)
```

Arguments

`x`	vector of counts.

Value

`min.KS`	The value of the KS statistic for the fitted Pareto distribution.
`Cmin`	The inferred threshold.
`powerlaw.exponent`	The powerlaw exponent. This is equal to `pareto.alpha + 1`
`pareto.alpha`	The inferred shape parameter alpha of the fitted Pareto distribution.

Author(s)

`<hbk5086@psu.edu>`

References

Desponds, Jonathan, Thierry Mora, and Aleksandra M. Walczak. "Fluctuating fitness shapes the clone-size distribution of immune repertoires." Proceedings of the National Academy of Sciences 113.2 (2016): 274-279. APA

Examples

```
# Fit the model to sample data

data("repertoires")

fit1 <- fdesponds(repertoires[[1]])
fit2 <- fdesponds(repertoires[[2]])

fit1
fit2
```

`fdiscgammagpd` *Fit the discrete gamma-GPD spliced threshold model*

Description

This function takes count data and fits the gamma-GPD spliced threshold model to it. The model consists of a discrete truncated gamma as the bulk distribution, up to the threshold, and a discrete GPD at and above the threshold. The 'shift' is ideally the minimum count in the sample.

Usage

```
fdiscgammagpd(x, useq, shift = NULL, pvector=NULL,
std.err = TRUE, method = "Nelder-Mead", ...)
```

Arguments

`x`	A vector of count data.
`useq`	A vector of possible thresholds to search over. These should be discrete numbers.
`shift`	The amount the distribution is shifted. It is recommended to use the minimum number in the count data when modeling the clone size distribution of the TCR repertoire.
`pvector`	A vector of 5 elements corresponding to the initial parameter estimates. These 5 initial values are for the gamma shape and rate, the threshold, and the GPD sigma and xi. If they are not prespecified, the function computes pvector automatically.
`std.err`	Logical. Should the standard errors on the estimates be computed from the Hessian matrix?
`method`	Character string listing optimization method fed to optim. Defaults to Nelder-Mead.
`...`	Other arguments passed to the function.

Value

`x`	Numerical vector of the original data input
`shift`	Numeric specifying the original shift input.
`init`	Numerical vector of the initial values of the parameter estimates. This is the same as pvector.
`useq`	Numerical vector containing the thresholds the grid search was performed over.
`nllhuseq`	Numerical vector of negative log likelihoods computed at each threshold in useq.
`optim`	Output from optim for the bulk and tail distributions.
`nllh`	The negative log likelihood corresponding to the maximum likelihood fitted distribution.
`mle`	A numerical vector containing the estimates for phi, shape, rate, threshold, sigma, and xi.
`fisherInformation`	The Fisher information matrix computed from the Hessian output from optim.

Author(s)

<hbk5086@psu.edu>

Examples

```
data("repertoires")
thresholds1 <- unique(round(quantile(repertoires[[1]], c(.75,.8,.85,.9,.95))))
thresholds2 <- unique(round(quantile(repertoires[[2]], c(.75,.8,.85,.9,.95))))

fit1 <- fdiscgammagpd(repertoires[[1]], useq = thresholds1,
                      shift = min(repertoires[[1]]))
fit2 <- fdiscgammagpd(repertoires[[2]], useq = thresholds1,
                      shift = min(repertoires[[2]]))

fit1
fit2
```

`get_bootstraps` *Get bootstrapped fits for a list of fitted models*

Description

In order to get confidence bands on parameter estimates, a parametric bootstrap is recommended. This bootstrapping procedure takes bootstraps above and below the threshold separately, retaining the correct proportion of data that are above or below the threshold.

Usage

```
get_bootstraps(fits, resamples = 1000, cores = 1, gridStyle = "copy")
```

Arguments

`fits`	A list of fits output from fdiscgammagpd.
`resamples`	Number of bootstrap replicates to execute for each model fit. Defaults to 1000.
`cores`	Number of cores to use, if running in parallel. Defaults to 1.
`gridStyle`	Defines how the sequence of thresholds is selected in the bootstrap fits. If the default "copy", each bootstrapped fit will be computed using the same grid of thresholds from the original fit. Otherwise, an integer can be supplied. If an integer is supplied, the bootstraps will be fit using a grid of thresholds defined by the originally estimated threshold plus or minus the supplied integer.

Value

If only one fit is passed, get_bootstraps returns a list of length resamples, where each element is a bootstrapped fit output from fdiscgammagpd. If a list of fits is passed, then the output is a list of lists. Each element of that list is a list of length resamples, where each element is a bootstrapped fit output from fdiscgammagpd.

Author(s)

`<hbk5086@psu.edu>`

See Also

`fdiscgammagpd`

Examples

```
data(repertoires)
fits <- lapply(repertoires,
           function(X) fdiscgammagpd(X, useq = unique(round(quantile(X, c(.75,.8,.85,.9))))))
names(fits) <- names(repertoires)

# You should in practice use a large number of resamples, say, 1000
boot <- get_bootstraps(fits, resamples = 10)

mles <- get_mle(boot[[1]])
xi_CI <- quantile(unlist(purrr::map(mles, 'xi')), c(.025, .5, .975))
xi_CI
```

`get_distances` *Compute matrix of pairwise Jensen-Shannon Distances*

Description

For a list of model fits (either the spliced model or the Desponds et al. model), compute the matrix of Jensen-Shannon distances. This can then be used for clustering or multi-dimensional scaling.

Usage

```
get_distances(fits, grid, modelType = "Spliced")
```

Arguments

`fits`	A list of fits output from either fdiscgammagpd or fdesponds.
`grid`	Vector of integers over which to compute the JS distance. The minimum of the grid is ideally the minimum count of all samples being compared. The maximum is ideally something very large (e.g. 100,000) in order to all or nearly all of the model density. The grid should include every integer in its range. See `JS_dist`.
`modelType`	Either "Spliced" or "Desponds", depending on what sort of fits you are supplying. Defaults to "Spliced".

Value

A symmetric matrix of pairwise Jensen-Shannon distances, with 0 on the diagonal.

Author(s)

`<hbk5086@psu.edu>`

See Also

`JS_dist`, `fdiscgammagpd`, `fdesponds`

Examples

```
# Simulate 3 datasets
set.seed(123)
s1 <- rdiscgammagpd(1000, shape = 3, rate = .15, u = 25, sigma = 15,
                    xi = .5, shift = 1)
s2 <- rdiscgammagpd(1000, shape = 3.1, rate = .14, u = 26, sigma = 15,
                    xi = .6, shift = 1)
s3 <- rdiscgammagpd(1000, shape = 10, rate = .3, u = 45, sigma = 20,
                    xi = .7, shift = 1)

# Fit the spliced model to each
# Here, we use true thresholds for fast computation for this example
# In practice, you need to select a whole sequence of potential thresholds
sim_fits <- list("s1" = fdiscgammagpd(s1, useq = 25),
                 "s2" = fdiscgammagpd(s2, useq = 26),
                 "s3" = fdiscgammagpd(s3, useq = 45))

# Compute the pairwise JS distance between 3 fitted models
grid <- min(c(s1,s2,s3)):10000
distances <- get_distances(sim_fits, grid, modelType="Spliced")
distances
```

`JS_desponds` *Compute the Jensen-Shannon distance between two fitted distributions using the Desponds model*

Description

After the Desponds et al. (2016) model havs been fit to your samples, the pairwise JS distance can be computed between them. This function takes the fitted model parameters from 2 distributions and computes the JS distance between them. When all pairwise distances have been computed, they can be used to do hierarchical clustering. This function assumes you denote one distribution as P and one as Q.

Usage

```
JS_desponds(grid, Cminp, Cminq, alphap, alphaq)
```

Arguments

`grid`	Vector of integers over which to compute the JS distance. The minimum of the grid is ideally the minimum count of all samples being compared. The maximum is ideally something very large (e.g. 100,000) in order to all or nearly all of the model density. The grid should include every integer in its range. See Examples.
`Cminp`	The estimated threshold for distribution P.
`Cminq`	The estimated threshold for distribution Q.
`alphap`	The estimated parameter alpha for distribution P.
`alphaq`	The estimated parameter alpha for distribution Q.

Details

For 2 discrete distributions P and Q, the Jensen-Shannon distance between them is

$JSD(P,Q) = \sqrt{.}5 * \int[P(t)logP(t)/M(t)] + \int[Q(t)logQ(t)/M(t)]dt$

where

$M(t) = .5 * (P(t) + Q(t)).$

Value

The function directly returns the Jensen-Shannon distance between two fitted distributions P and Q.

Author(s)

`<hbk5086@psu.edu>`

References

Desponds, Jonathan, Thierry Mora, and Aleksandra M. Walczak. "Fluctuating fitness shapes the clone-size distribution of immune repertoires." Proceedings of the National Academy of Sciences 113.2 (2016): 274-279. APA

See Also

`JS_spliced`, `JS_dist`

Examples

```
data("repertoires")

# Fit the discrete gamma-gpd spliced model at some selected threshold on 2 samples
fit1 <- fdesponds(repertoires[[1]])
fit2 <- fdesponds(repertoires[[2]])

# Create a grid of every integer from the minimum threshold to a large value
# When comparing many distributions in advance of clustering,
# the same grid should be used across every comparison
# The chosen "large value" here is only 1,000, for the sake of quick computation.
# Ideally, the large value will be at least 100,000
grid <- min(c(fit1['Cmin'], fit2['Cmin'])):1000

# Compute the Jensen-Shannon distance between fit1 and fit2
dist <- JS_desponds(grid,
                    Cminp = fit1['Cmin'],
                    Cminq = fit2['Cmin'],
                    alphap = fit1['pareto.alpha'],
                    alphaq = fit2['pareto.alpha'])

dist
```

`JS_dist` *Compute the Jensen-Shannon distance between two model fits*

Description

This function is a convenient wrapper for JS_spliced and JS_desponds. After models have been fit to your samples, the pairwise JS distance can be computed between them. This function takes two model fits and outputs the JS distance between them. The model fits must be of the same type. That is, they are both fits from the spliced threshold model, or they are both fits from the Desponds et al. model. When all pairwise distances have been computed, they can be used to do hierarchical clustering.

Usage

```
JS_dist(fit1, fit2, grid, modelType = "Spliced")
```

Arguments

`fit1`	A fit from the specified modelType.
`fit2`	A fit from the specified modelType.
`grid`	Vector of integers over which to compute the JS distance. The minimum of the grid is ideally the minimum count of all samples being compared. The maximum is ideally something very large (e.g. 100,000) in order to all or nearly all of the model density. The grid should include every integer in its range. See Examples.
`modelType`	The type of model fit1 and fit2 are from. If they were generated using fdiscgammagpd, the type of model is "Spliced". If they were generated using fdesponds, the type of model is "Desponds". Defaults to "Spliced".

Details

For 2 discrete distributions P and Q, the Jensen-Shannon distance between them is

$JSD(P,Q) = \sqrt{.}5 * [\sum(P_i log P_i/M_i)] + \sum(Q_i log Q_i/M_i)$

where

$M_i = .5 * (P_i + Q_i)$.

Value

The function directly returns the Jensen-Shannon distance between two fitted distributions.

Author(s)

<hbk5086@psu.edu>

See Also

JS_spliced, JS_desponds

Examples

```
data("repertoires")

# Fit the discrete gamma-gpd spliced model at some selected threshold on 2 samples
fit1 <- fdiscgammagpd(repertoires[[1]],
                        useq = quantile(repertoires[[1]], .8),
                        shift = min(repertoires[[1]]))
fit2 <- fdiscgammagpd(repertoires[[2]],
                        useq = quantile(repertoires[[2]], .8),
                        shift = min(repertoires[[2]]))

# Create a grid of every integer from the minimum count to a large value
# The chosen "large value" here is only 1,000, for the sake of quick computation.
# Ideally, the large value will be at least 100,000
grid <- min(c(repertoires[[1]], repertoires[[2]])):1000

# Compute the Jensen-Shannon distance between fit1 and fit2
dist <- JS_dist(fit1, fit2, grid, "Spliced")
dist
```

`JS_spliced` *Compute the Jensen-Shannon distance between two fitted discrete gamma-GPD spliced threshold distributions*

Description

After models have been fit to your samples, the pairwise JS distance can be computed between them. This function takes the fitted model parameters from 2 distributions and computes the JS distance between them. When all pairwise distances have been computed, they can be used to do hierarchical clustering. This function assumes you denote one distribution as P and one as Q.

Usage

```
JS_spliced(grid, shiftp, shiftq, phip, phiq, shapep, shapeq,
           ratep, rateq, threshp, threshq, sigmap, sigmaq, xip, xiq)
```

Arguments

`grid`	Vector of integers over which to compute the JS distance. The minimum of the grid is ideally the minimum count of all samples being compared. The maximum is ideally something very large (e.g. 100,000) in order to all or nearly all of the model density. The grid should include every integer in its range. See Examples.
`shiftp`	The shift for distribution P.
`shiftq`	The shift for distribution Q.
`phip`	The estimated phi for distribution P.
`phiq`	The estimated phi for distribution Q.
`shapep`	The estimated gamma shape parameter for distribution P.
`shapeq`	The estimated gamma shape parameter for distribution Q.
`ratep`	The estimated gamma rate parameter for distribution P.
`rateq`	The estimated gamma rate parameter for distribution Q.
`threshp`	The estimated threshold for distribution P.
`threshq`	The estimated threshold for distribution Q.
`sigmap`	The estimated parameter sigma for distribution P.
`sigmaq`	The estimated parameter sigma for distribution Q.
`xip`	The estimated parameter xi for distribution P.
`xiq`	The estimated parameter xi for distribution Q.

Details

For 2 discrete distributions P and Q, the Jensen-Shannon distance between them is

$JSD(P,Q) = \sqrt{.}5 * [\sum(P_i log P_i/M_i)] + \sum(Q_i log Q_i/M_i)$

where

$M_i = .5 * (P_i + Q_i)$.

Value

The function directly returns the Jensen-Shannon distance between two fitted distributions P and Q.

Author(s)

`<hbk5086@psu.edu>`

See Also

`JS_spliced` `JS_dist`

Examples

```
data("repertoires")

# Fit the discrete gamma-gpd spliced model at some selected threshold on 2 samples
fit1 <- fdiscgammagpd(repertoires[[1]],
                        useq = quantile(repertoires[[1]], .8),
                        shift = min(repertoires[[1]]))
fit2 <- fdiscgammagpd(repertoires[[2]],
                        useq = quantile(repertoires[[2]], .8),
                        shift = min(repertoires[[2]]))

# Create a grid of every integer from the minimum count to a large value
# The chosen "large value" here is only 1,000, for the sake of quick computation.
# Ideally, the large value will be at least 100,000
grid <- min(c(repertoires[[1]], repertoires[[2]])):1000

# Compute the Jensen-Shannon distance between fit1 and fit2
dist <- JS_spliced(grid,
                    shiftp = min(repertoires[[1]]),
                    shiftq = min(repertoires[[2]]),
                    phip = fit1$mle['phi'], phiq = fit2$mle['phi'],
                    shapep = fit1$mle['shape'], shapeq = fit2$mle['shape'],
                    ratep = fit1$mle['rate'], rateq = fit2$mle['rate'],
                    threshp = fit1$mle['thresh'], threshq = fit2$mle['thresh'],
                    sigmap = fit1$mle['sigma'], sigmaq = fit2$mle['sigma'],
                    xip = fit1$mle['xi'], xiq = fit2$mle['xi'])

dist
```

`parseFile` *Load in and parse TCR data files for use by powerTCR*

Description

This function leverages repLoad functions from the immunarch package. They are wrappers that output data in the format taken by powerTCR.

Usage

```
parseFile(path, inframe = TRUE)
```

Arguments

`path`	Path to input file with TCR repertoire sample data.
`inframe`	Logical. Should counts only from in-frame sequences be returned? Defaults to TRUE.

Details

See the immunarch package on GitHub: https://github.com/immunomind/immunarch

Value

`parseFile` returns a vector of counts corresponding to the sample repertoire.

References

Nazarov, Vadim I., et al. "tcR: an R package for T cell receptor repertoire advanced data analysis." BMC bioinformatics 16.1 (2015): 175.

`repertoires` *Two toy examples of sample TCR repertoires.*

Description

This data set gives to toy examples of TCR repertoires. Sample "samp1" contains 1,000 clones with a total of 26,288 sequenced T cells. Sample "samp2" contains 800 clones with a total of 24,267 sequenced T cells. These samples have been sorted from largest to smallest clone size.

Usage

```
data("repertoires")
```

Format

The format is:

List of 2 $ samp1: num [1:1000] 1445 451 309 ... $ samp2: num [1:800] 2781 450 447 ...

Examples

```
data(repertoires)
n1 <- length(repertoires$samp1)
n2 <- length(repertoires$samp2)

# Generates plot on log-log scale
par(mfrow = c(2,1))
plot(log(repertoires$samp1), log(1:n1))
plot(log(repertoires$samp2), log(1:n2))
```

Index