Package ‘GeneticTools’

February 19, 2015

Type Package
Title Collection of Genetic Data Analysis Tools
Version 0.3.1
Date 2015-02-04
Author Daniel Fischer
Maintainer Daniel Fischer <daniel.fischer@luke.fi>
Depends R (>= 3.0.2), gMWT (>= 0.3), snpStats, Rcpp (>= 0.9.13), plotrix
LinkingTo Rcpp, RcppArmadillo
Description A loose collection of tools for the analysis of gene expression and genotype data, currently with main focus on eQTL and MDR analysis.
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-02-04 13:12:25

R topics documented:

GeneticTools-package ................................................. 2
annotTrack .......................................................... 2
calcDistances ....................................................... 3
eQTL ................................................................. 4
genotData ............................................................ 7
genotypePlot ......................................................... 7
getRegionsOI ........................................................ 9
gtfToBed ............................................................. 9
mdr ................................................................. 10
plot.eqtl ............................................................ 12
plot.mdr ............................................................ 13
predict.mdr ......................................................... 14
print.eqtl ........................................................... 15
print.mdr ........................................................... 16
Description

A loose collection of analysis tools for gene expression and genotype data. Currently it provides methods for eQTL, QTL and MDR analysis. It also contains an MDR ensemble classifier.

Details

Package: GeneticTools
Type: Package
Version: 0.3.1
Date: 2015-02-04
License: GPL
LazyLoad: yes

Author(s)

Daniel Fischer
Maintainer: Daniel Fischer <daniel.fischer@luke.fi>

Example Annotation Track

Description

Example annotation track.

Usage

data(annotTrack)
**Format**

A data frame with 1001 rows, each representing one annotation with 9 columns as provided from standard gtf format.

**Details**

This is an example homo sapiens annotation track as it was imported from the Ensemble ftp download page. In total there are 1001 annotations from the human genome in release 70.

**Source**

http://www.ensembl.org/info/data/ftp/index.html

**Examples**

data(annotTrack)
head(annotTrack)

calcDistances

---

**Description**

This is an experimental function to calculate disances between time series.

**Usage**

calcDistances(X, method="C", nodes=20)

**Arguments**

- **X**: Matrix with time series values.
- **method**: Method of choice to calculate the distance.
- **nodes**: Number of splines for the spline-based approach.

**Details**

This is an early attempt to implement a distance measure for time series clustering. It has not yet been tested and hence should be used carefully.

Currently there are two different methods available, the default is `method="C"`. The alternative is `spline`, what is a spline based approach.

The matrix X contains the individuals in the rows and the columns represent the time points.

As this is developing code, we do not give further information, yet, but include the function in the package for testing purposes.
**Value**

A matrix with pairwise distances.

**Author(s)**

Daniel Fischer

---

**eQTL**

*Perform an eQTL Analysis*

**Description**

This function performs an eQTL analysis.

**Usage**

```r
eqtl(gex, geno, xAnnot = NULL, xSamples = NULL, genoSamples = NULL,
     windowSize = 0.5, method = "LM", mc = 1, sig = NULL, which = NULL,
     nper = 2000, verbose = TRUE)
```

**Arguments**

- **gex**: Matrix or Vector with expression values.
- **geno**: Genotype data.
- **xAnnot**: Location annotations for the expression values.
- **xSamples**: Sample names for the expression values, see details (optional).
- **genoSamples**: Sample names for the genotype values, see details (optional).
- **windowSize**: Size of the window around the center gene, see details.
- **method**: Method of choice for the eQTL, see details.
- **mc**: Amount of cores for parallel computing.
- **sig**: Significance level for the eQTL testing, see details.
- **which**: Names of genes for which the eQTL should be performed.
- **nper**: Sets the amount of permutations, if permutation tests are used.
- **verbose**: Logical, if the method should report intermediate results.

**Details**

This function performs an eQTL analysis and offers different types of tests. The type of test can be specified with the `method` option and possible options are "LM" and "directional". The option "LM" fits for each SNP within a predefined window of size `windowSize` (in MB) around a gene a linear model for the genotype information and the corresponding gene expression. The null hypothesis for each test is then that the slope is equal to zero and the alternative is that it is not zero.

The "directional" option applies a new directional test based on probabilistic indices for triples as described in Fischer, Oja, et al. (2013). Being $x_0 = (x_{01}, x_{02}, \ldots, x_{0N_0})'$, $x_1 = (x_{11}, x_{12}, \ldots, x_{1N_1})'$
eQTL

and \(x_2 = (x_{21}, x_{22}, \ldots, x_{2N_2})'\) the expression values that are linked to the three genotype groups 0, 1 and 2 with underlying distributions \(F_0, F_1\) and \(F_2\). We first calculate the probabilistic indices \(P_{0,1,2} = \frac{1}{N_0N_1N_2} \sum_{i} \sum_{j} I(x_{0i} < x_{1j} < x_{2k})\) and \(P_{2,1,0} = \frac{1}{N_0N_1N_2} \sum_{i} \sum_{j} \sum_{k} I(x_{2i} < x_{1j} < x_{0k})\). These are the probabilities that the expression values of the three groups follow a certain order what we would expect for possible eQTLs. The null hypothesis that we have then in mind is that the expression values from these three group have the same distribution \(H_0 : F_0 = F_1 = F_2\) and the two alternatives are that the distributions have a certain stochastic order \(H_1 : F_0 < F_1 < F_2\) and \(H_2 : F_2 < F_1 < F_0\).

The test is applied for the two probabilistic indices \(P_{0,1,2}\) and \(P_{2,1,0}\) and combines the two resulting p-values \(p_{012} = p_1\) and \(p_{210} = p_2\) from previous tests then as overall p-value \(\min(2 \min(p_1, p_2), 1)\).

In the two-group case (this means only two different genotypes are present for a certain SNP) a two-sided Wilcoxon rank-sum test is applied.

The gene expressions are specified in gex. If several genes should be tested, then gex is a matrix and each column refers to a gene and each row to an individual. The column names of this matrix should match then with the names used in the annotation object xannot. Sample names can either be given as row names in the matrix or as separate vector in xsamples. If only gene expressions of one gene should be tested then gex can be a vector.

The genotype information is provided in the geno object. Here one can either specify the file name of a ped/map file pair. In that case the function imports the genotype information using the SnpStats package. In case the genotype information has been imported already earlier using SnpStats::read_pedfile() the resulting SnpMatrix can also be given as a parameter for geno.

The xannot object carries the annotation information for the gene expressions. In case of multiple locations per gene it is of type list and each list item stores the information for one gene in form of a data.frame in bed format. This data.frame has then the three columns Chr, Start, End and each row refers to one matching chromosomal position of the underlying gene. Especially when probes of ssRNAs are considered the chromosomal positions of a probe are not necessarily unique. The names of the list xannot are the names of the genes and they have to match with the column names of gex. However, the order does not have to be the same, and xannot can include more annotations of genes than given in gex. The function finds and uses then the union between the column names of gex and the list entries of xannot. Alternative xannot can also be a data frame if unique locations are considered. In that case xannot has to be a data frame with the four columns Gene, Chr, Start, End.

The option genoSamples is used in case that the sample names in the ped/map file (or SnpMatrix) do not match with rownames(gex) given in the expression matrix. The vector genoSamples is as long as the geno object has samples, but gives then for each row in geno the corresponding name in the gex object. The function finds then also the smallest union between the two data objects. If there are repeated measurements per individual for the genotypes we take by default only the first appearance in the data and neglect all successive values. Currently this cannot be changed. In case this behavior is not desired, the user has to remove the corresponding rows from geno before starting the calculation.

If the code is executed on a Linux OS the user can specify with the mc option the amount of CPU cores used for the calculation.

If the sig option is set to a certain significance level, then the method only reports those SNPs that are tested to be significant. This can reduce the required memory drastically, especially in the case of trans-eQTL.
The method tests for trans-eQTLs (all combinations of SNPs and genes) if the windowSize is set to 0 or NULL. Be aware that this might lead to long lasting calculations.

Note: The directional test currently supports only exact p-values based on permutation tests, but asymptotic implementations are developed and will be soon available also.

Value
A list of class eqtl containing the values

- gex: The gex object from the function call.
- geno: The geno object from the function call.
- xAnnot: The xAnnot object from the function call.
- genoSamples: The genoSamples object from the function call.
- windowSize: The windowSize object from the function call.

and an incapsulated list eqtl where each list item is a tested gene location and contains the items

- ProbeLoc: Used position of that gene. (Only different from 1 if multiple locations are considered.)
- TestedSNP: Details about the considered SNPs.
- p.values: P values of the test.
- GeneInfo: Details about the center gene.

Author(s)
Daniel Fischer

References

Examples

# Please, see also the package vignette for a more descriptive example section on this.

# Make the example data available
data(xgene)
data(genotData)
data(annotTrack)

# We need to have the gene annotation in bed format (Please notice the change to the
# official convention, this is on high priority of the ToDo list of the package to change
# this.)

## Not run:
### genotData

**Simulated Genotype Data**

**Description**

Simulated genotype data

**Usage**

```r
data(genotData)
```

**Format**

A list as provided from `snpStats::read_pedfile()`

**Details**

Small simulated genotype dataset that can be used for testing the method.

**Source**

Own simulation, code can be given upon request.

**Examples**

```r
data(genotData)
```

### genotypePlot

**Expression Boxplots**

**Description**

Expression values are grouped according to genotype groups and then visualized with boxplots.

**Usage**

```r
genotypePlot(snp, gene, eqtl, geneAnnot=NULL, ylab=NULL, xlab=NULL, mainlab=FALSE)
```
Arguments

- `snp` String, specifies the genotype name.
- `gene` String, specifies the gene name.
- `eqtl` An eqtl object.
- `geneAnnot` String, specifies the gene name.
- `ylab` Optional x-axis label.
- `xlab` optional y-axis label
- `mainlab` Logical, shall main title be plotted

Details

This function plots the expression values of the genotype groups of a certain SNP that can be given in the `snp` option. The expression values are specified in the `gene` option.

The `eqtl` object is the output of an eQTL run and carries the required genotype information.

Value

A Figure.

Author(s)

Daniel Fischer

Examples

```r
# Make the example data available
data(Xgene)
data(genotData)
data(annotTrack)

# We need to have the gene annotation in bed format (Please notice the change to the official convention, this is on high priority of the ToDo list of the package to change this.)
## Not run:
  annotBed <- gtfToBed(annotTrack)

# Perform a basic cis-eQTL with the minimum required input linear model:
  lm.myEQT <- eQTL(gex=Xgene, geno=genotData, xAnnot=annotBed, method="LM", windowsize=1)

# Plot the genotypes
  genotypePlot(snp="SNP377", gene="MYBPC1", eqtl=lm.myEQT, ylab="Expression values", xlab="Genotypes")
```

## End(Not run)
## getRegionsOI

**Filter Annotation File**

### Description
Filter out unimportant regions of an annotation file in bed format.

### Usage
```r
getRegionsOI(annot, regOI)
```

### Arguments
- **annot**: An annotation file in bed format.
- **regOI**: A dataframe specifying the interesting location.

### Details
When considering trans-eQTLs all combinations between gene annotations and genotype data are tested. This often forces the user to filter the annotations to a certain region. Having an annotation file in bed format with the columns `name, chr, start, and stop` and a data frame specifying the different locations that should remain, this function removes all annotations outside these given areas. The column names of the data frame given to `regOI` are `chr, start, and end`.

### Value
A subset of `annot`.

### Author(s)
Daniel Fischer

## gtfToBed

**Extract the Chromosomal Information Required in bed Format from an imported gtf table.**

### Description
This function creates a matrix of gene annotations in bed format, based on the information given in an imported gtf table.

### Usage
```r
gtfToBed(gtf, output="min")
```
Arguments

- **gtf**: An imported gtf table.
- **output**: Option to format the output, see details.

Details

Currently the function supports only gtf files for human organisms. If applied to other organisms the code has to be changed in such a way that the Chromosome names are adjusted. The output can be formatted with the output option, where `output="min"` results in one row per gene and `output="full"` keeps the original format in the gtf file.

Value

A data.frame in bed format having the four columns `Chr`, `Start`, `Stop` and `Name`

Author(s)

Daniel Fischer

Examples

```r
## Not run: read.table(file="Homo_sapiens.GRCh37.70.gtf",sep="\t")
annotBed <- gtfToBed(annotTrack)
## End(Not run)
```

---

**mdr**

*Perform a MDR.*

Description

This function performs a Multifactor Dimension Reduction (MDR).

Usage

```r
mdr(X,status,fold=2,t=NULL,cv=0,cvp=0.75,top=20,NAasValues=TRUE,fix=NULL)
```

Arguments

- **X**: Matrix with genotype information, see details.
- **status**: Vector with group information of individuals in X.
- **fold**: Maximum dimension of used contingency tables, see details.
- **t**: Threshold for high/low risk.
- **cv**: Amount of cross validation runs.
- **cvp**: Ratio of cross-validation sample.
top  Length of each top list.
NAasValues  How shall NAs be treated.
fix  Shall one genotype be fixed.

Details
The matrix \( X \) contains the genotype information or the filename of a ped/map filepair. If a ped/map filename is given the status information from this pair is taken and no further status object has to be given. In case \texttt{status} is given as well, we will take this information.

In case the matrix \( X \) is not given in 0,1,2 format the function \texttt{recodeData} recodes the data into the required 0,1,2 format.

The \texttt{status} vector is as long as \( X \) has individuals and specifies the group labels for each individual. Healthy individual shall be encoded as 0 and cases as 1. If the labeling is different we take the smaller values as controls and the larger one as cases.

The \texttt{fold} option specifies up to which dimension the contingency tables should be used. The current maximum is four.

The \texttt{t} option gives the threshold for the classification between high and low risk classes. The default is the ratio of the groups sizes.

Value
An object of class \texttt{mdr}.

Author(s)
Daniel Fischer

References

Examples
# The datasets are not yet available. As soon as they will get published they will
# be also added to the package

## Not run:
# Read in the genotype data
genotData <- read.table("MDR_data.txt",header=T)

# Extract the status information
status <- genotData[,"Class"]
genotData <- as.matrix(genotData[-which((colnames(genotData)=="Class")==TRUE)])

# Bring the data into 0,1,2 format
temp <- recodeData(genotData)
# Perform the MDR
res <- mdr(x=temp,status=status,fold=3,top=20)

## End(Not run)

---

### plot.eqtl

**Plot an eqtl Object**

**Description**

The function offers informative plots for an eqtl object.

**Usage**

```r
## S3 method for class 'eqtl'
plot(x, file = NULL, which = NULL, sig = 0.01, verbose = TRUE, centered = TRUE,
     log = FALSE, x2 = NULL, annot = NULL, track = NULL,
     trackAnnot = FALSE, trackOrder = NULL, mc.cores = 1, ...)
```

**Arguments**

- `x` Object of class eqtl.
- `file` Store set of graphics under that file name.
- `which` Specifies for which genes should the plot be created.
- `sig` Chosen significance level.
- `verbose` Logical, extended feedback of the function.
- `centered` Logical, plot should be centered around center gene.
- `log` Logical, y-axis scale is log(base=10)-scaled.
- `x2` Comparison values of a second eqtl object, see details.
- `annot` Logical, plot annotation track.
- `track` Gene annotations in bed format.
- `trackAnnot` Gene annotations in bed format.
- `trackOrder` Logical, shall the annotation track be ordered.
- `mc.cores` Amount of cores for parallel computing.
- `...` Additional plotting parameters.

**Details**

This function plots the test results of an eqtl object. Typically is the tested gene in the center and the p-values of associated SNPs are visualized. Monomorphic SNPs and those that were missing are separately plotted. Test results that are smaller than the value given to `sig` are marked in red. The y-axis can be switched to log10 scale by setting the logical parameter `log=TRUE` in that case are bars instead of dots plotted. If the y-axis is on log-scale it is also possible to give a second eqtl object to the function and plot the test results for both.

The annotation feature is currently under development and only available in limited form.
Author(s)
Daniel Fischer

Examples

```r
# Perform eQTL (single location, one gene):
data(xgene)
data(genoData)
data(annotTrack)
## Not run:
annotBed110 <- gtfToBed(annotTrack[1:10,])

lm.myEQTL <- eQTL(gex=xgene, geno=genoData, xAnnot=annotBed110, method="LM")
plot(lm.myEQTL)

dir.myEQTL <- eQTL(gex=xgene, geno=genoData, xAnnot=annotBed110, method="directional")
plot(lm.myEQTL, x2=dir.myEQTL, log=TRUE, sig=2)
## End(Not run)
```

Description

The function offers informative plots for an mdr object.

Usage

```r
## S3 method for class 'mdr'
plot(x, which=NULL, ...)
```

Arguments

- `x` Object of class mdr.
- `which` Specifies for which association fold should the plot be created.
- `...` Additional plotting parameters.

Details

This function plots the density of the precision of an mdr object.

Author(s)
Daniel Fischer
predict.mdr

**Examples**

```r
# Perform the MDR
# res <- mdr(X=temp,status=status,fold=3,top=20)
# plot(res)
```

**Description**

This is an mdr ensemble classifier.

**Usage**

```r
## S3 method for class 'mdr'
predict(object, data=NULL, status=NULL, fold=NULL, ...)
```

**Arguments**

- **object**: Object of class mdr.
- **data**: The new data object.
- **status**: Optional, used for 2x2 classification table.
- **fold**: Considered dimension of the model.
- **...**: Additional parameters

**Details**

Given an mdr object this function takes the top list for the (highest) fold group and uses it as an ensemble classifier for the new data given in the data argument.

**Value**

A vector, giving for each subject from the new data object a classification.

**Author(s)**

Daniel Fischer

**Examples**

```r
# indices <- 1:nrow(genotData)
# trainSet <- sample(indices,100)
# testSet <- indices[-trainSet]
# temp <- recodeData(genotData)
# res <- mdr(X=temp[trainSet,], status=status[trainSet], fold=3, top=20)

# trainRes <- predict(res, data=temp[trainSet,])
# testRes <- predict(res, data=temp[testSet,])
```
print.eqtl  

Print an eqtl Object

Description

Prints an eqtl object.

Usage

```r
## S3 method for class 'eqtl'
print(x, which=NULL, sig=0.01, output="bed", ...)
```

Arguments

- `x`: Object of class eqtl.
- `which`: Which center gene should be printed.
- `sig`: Significance level.
- `output`: Output format.
- `...`: Additional parameters.

Details

The function prints SNPs in the surroundings of a gene from an eqtl object.

By default all genes are considered, subsets can be defined with the `which` option. The `sig` option gives the threshold which results should be shown.

Author(s)

Daniel Fischer

Examples

```r
## Not run:
myeqtl <- eQTL(geneMatrix, genoData, singleLoc, genoSamples, singleSamples, windowSize, method="LM")
myeqtl
print(myeqtl, sig=0.05)

## End(Not run)
```
print.mdr  

*Print an mdr Object*

**Description**

Prints an `mdr` object.

**Usage**

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

**Arguments**

- `x` Object of class `mdr`.
- `...` Additional parameters.

**Details**

The function prints an `mdr` object.

**Author(s)**

Daniel Fischer

**Examples**

```r
# res <- mdr(X=temp,status=status,fold=3,top=20)
# res
```

---

realSNPs  

*Simulated SNP Data*

**Description**

Simulated SNP data.

**Usage**

```r
data(realSNPs)
```

**Format**

Data format as it is provided from package::snpStats and its function `read.pedfile`.
recodeData

Details
Just simulated data for testing purposes of the functions.

Source
Own imulation, underlying code can be given upon request.

Examples
data(realSNPs)

---

**recodeData**  
*Recode a Genotype Data Matrix.*

Description
This function recodes a genotype data matrix in a format as expected from eqtl or mdr.

Usage
recodeData(x)

Arguments

- **x**  
  Matrix with genotype information.

Details
This function recodes the values given in the data matrix `x` (typically AA, AB and BB) and substitutes it with 0, 1 and 2. Missing values are encoded as 3.

Value
A matrix with same dimension as the input, but in 0,1,2 encoding.

Author(s)
Daniel Fischer

See Also
eQTL, mdr

Examples
# genotData <- read.table("MDR_format_ready_BCR.txt",header=T)
# temp <- recodeData(genotData)
summary.eqtl  

Summarize an eqtl Object

Description
Summarizes and prints an eqtl object in an informative way.

Usage
```r
## S3 method for class 'eqtl'
summary(object, ...)
```

Arguments
- `object`: Object of class eqtl.
- `...`: Additional parameters.

Details
This function gives a summary of an eqtl object.

Author(s)
Daniel Fischer

Examples
```r
# First perform an eQTL
# lm.myEQTL <- eQTL(gex=ourExpression, geno=genotData, xAnnot=xAnnotDF, windowSize=1,
#                 genoSamples=genoSamples, method="LM")

# summary(lm.myEQTL)
```

visTrans  

Visualize trans-eQTL Results

Description
Plot function for the visualization of trans-eQTL results.

Usage
```r
visTrans(snpGene, geneAnnot)
```
Arguments

snpGene A dataframe indicating the snp-gene association, see details.
geneAnnot An annotation track.

Details

Typically are trans-eQTL difficult to visualize. One possible option is this plot. It takes as an input a dataframe in bed format. This one is typically provided from an eQTL run with set parameter sig in the bed list. The dataframe indicates then all significant associations between genes and SNPs and these associations are then connected within the Figure with arches.

In addition it is possible to plot an annotation track, by specifying it in geneAnnot. This track is required to be in bed format and the column names are Name, Chr, Start and Stop. In case a standard Ensemble gtf file is used for that the function gtftoBed provides the correct input for this option.

Value

A figure.

Author(s)

Daniel Fischer

Examples

# This is just simulated data and hence we cannot see results as with real data.
# An example for real data is shown in the vignette and as soon as this data is
# freely available this example will be updated.

# See also the vignette for a more detailed example.

data(Xgene)
data(genoData)
data(annotTrack)

# Not run:
annotBed <- gtftoBed(annotTrack)

lm.myEQTltrans <- eQTL(gex=Xgene, geno=genoData, xAnnot=annotBed,
method="LM", windowSize=NULL, sig=0.01)

snpGeneInfo <- lm.myEQTltrans$bed

# Plot the visualization
visTrans(snpGene=snpGeneInfo,annotBed)

## End(Not run)
Description

Simulated expression data.

Usage

\texttt{data(Xgene)}

Format

A matrix with 100 rows representing individuals and 13 genes arranged in the columns.

Details

Just simulated, numerical expression data for testing purposes.

Source

Own simulation, code can be given upon request.

Examples

\texttt{data(Xgene)}
Index

+Topic **datasets**
  - annotTrack, 2
  - genotData, 7
  - realSNPs, 16
  - Xgene, 20
+Topic **hplot**
  - genotypePlot, 7
  - plot.eqtl, 12
  - plot.mdr, 13
  - visTrans, 18
+Topic **methods**
  - calcDistances, 3
  - eQTL, 4, 17
  - GeneticTools-package, 2
  - genotData, 7
  - genotypePlot, 7
  - getRegionsOI, 9
  - gtfToBed, 9
  - mdr, 10, 17
  - plot.eqtl-method (plot.eqtl), 12
  - plot.mdr-method (plot.mdr), 13
  - plot-method (plot.eqtl), 12
  - plot.eqtl, 12
  - plot.mdr, 13
  - predict.mdr, 14
  - print.eqtl, 15
  - print.mdr, 16
  - summary.eqtl, 18
+Topic **multivariate**
  - GeneticTools-package, 2
  - gtfToBed, 9
  - mdr, 10
  - recodeData, 17
+Topic **print**
  - print.eqtl, 15
  - print.mdr, 16
  - summary.eqtl, 18
+Topic **calcDistances, 3**
  - eQTL, 4
  - genotypePlot, 7
  - getRegionsOI, 9
  - gtfToBed, 9
  - mdr, 10
  - recodeData, 17
  - visTrans, 18
  - annotTrack, 2