Package 'tornado'

September 5, 2012

Type Package

Title Differential expression analysis of per-nucleotide coverage tables.

Version 1.0

Date 2012-08-27

Author Alyssa Frazee

Maintainer Alyssa Frazee <afrazee@jhsph.edu>

Description Creates SQLite database of per-nucleotide coverage files, fits linear model to each nucleotide to determine differential expression, fits Hidden Markov Model using moderated t statistics from linear models as state emissions to get a list of differentially expressed regions, connects found regions with known annotation.

License None yet.

R topics documented:

`tornado-package` *Tornado*

Description

Differential expression analysis of per-nucleotide coverage tables.

Details

Package:	tornado
Type:	Package
Version:	1.0
Date:	2012-08-22
License:	What license is it under?

Creates SQLite database of per-nucleotide coverage files (`makeDb`), fits linear model to each nucleotide to determine differential expression (`getLimmaInput`,`getTstats`), fits Hidden Markov Model using moderated t statistics from linear models as state emissions to get a list of differentially expressed regions (`getRegions`), connects found regions with known annotation (`getAnnotation`, `plotRegion`, `plotExon`, `plotGene`).

Author(s)

Alyssa Frazee <afrazee@jhsph.edu>

References

Paper coming soon.

`find.mean` *optional helper function for getParams*

Description

finds mean of distribution of either over- or underexpressed statistics.

Usage

```
find.mean(init.value, null.mean, null.sd, null.prop, vals, up = TRUE)
```

Arguments

`init.value`	number in (0, 0.5) representing a percentile of the null distribution to use as starting value
`null.mean`	estimated mean of null distribution (usually found with locfdrFit)
`null.sd`	estimated standard deviation of null distribution (usually found with locfdrFit)
`null.prop`	estimated proportion of statistics that came from the null distribution
`vals`	vector of all the observed values from the mixture distribution
`up`	if TRUE, find mean of overexpressed statistics, otherwise find mean of underexpressed statistics

Details

For experienced users/debugging only. Calls `find.mean.up` if up=TRUE, else calls `find.mean.down`. Most users should use `getParams` rather than `find.mean`.

Value

If numerical method succeeds, a list with elements

`m`	estimated mean of the underexpressed distribution, and
`p`	the percentile of the null distribution used to find this mean

If numerical method fails, a list with elements

`m`	mean of underexpressed distribution, as estimated by the 5th percentile of the estimated null distribution
`s`	standard deviation of underexpressed distribution, as estimated by the standard deviation of the null distribution

Author(s)

Alyssa Frazee

See Also

`getParams`, `find.mean.up`, `find.mean.down`

`find.mean.down` *helper function for getParams*

Description

finds mean of the distribution of statistics generated from underexpressed nucleotides

Usage

```
find.mean.down(init.value, null.mean, null.sd, null.prop, vals)
```

Arguments

`init.value`	number in (0, 0.5) representing a percentile of the null distribution to use as starting value
`null.mean`	estimated mean of null distribution (usually found with locfdrFit)
`null.sd`	estimated standard deviation of null distribution (usually found with locfdrFit)
`null.prop`	estimated proportion of statistics that came from the null distribution
`vals`	vector of all the observed values from the mixture distribution

Details

This function is for experienced users or debugging only - all other users should use `getParams`, which calls this function.

Value

If numerical method succeeds, a list with elements

`m`	estimated mean of the underexpressed distribution, and
`p`	the percentile of the null distribution used to find this mean

If numerical method fails, a list with elements

`m`	mean of underexpressed distribution, as estimated by the 5th percentile of the estimated null distribution
`s`	standard deviation of underexpressed distribution, as estimated by the standard deviation of the null distribution

Author(s)

Alyssa Frazee

See Also

`getParams`, `find.mean.up`, `find.sd`

`find.mean.up` *helper function for getParams*

Description

finds mean of the distribution of statistics generated from overexpressed nucleotides

Usage

```
find.mean.up(init.value, null.mean, null.sd, null.prop, vals)
```

Arguments

`init.value`	number in (0.5, 1) representing a percentile of the null distribution to use as starting value
`null.mean`	estimated mean of null distribution (usually found with locfdrFit)
`null.sd`	estimated standard deviation of null distribution (usually found with locfdrFit)
`null.prop`	estimated proportion of statistics that came from the null distribution
`vals`	vector of all the observed values from the mixture distribution

Details

This function is for experienced users or debugging only - all other users should use `getParams`, which calls this function.

Value

If numerical method succeeds, a list with elements

`m`	estimated mean of the underexpressed distribution, and
`p`	the percentile of the null distribution used to find this mean

If numerical method fails, a list with elements

`m`	mean of underexpressed distribution, as estimated by the 5th percentile of the estimated null distribution
`s`	standard deviation of underexpressed distribution, as estimated by the standard deviation of the null distribution

Author(s)

Alyssa Frazee

See Also

`getParams`, `find.mean.up`, `find.sd`

`find.sd` *helper function for getParams*

Description

find standard deviation of distribution of over- or underexpressed statistics

Usage

```
find.sd(prev.p, found.mean, null.mean, null.sd, null.prop, vals, up = T)
```

Arguments

prev.p	percentile of null distribution used to find the mean of the distribution of interest - usually the $p return of find.mean.up or find.mean.down.
found.mean	mean of distribution of interest - usually the $m return of find.mean.up or find.mean.down
null.mean	estimated mean of null distribution (usually found with locfdrFit)
null.sd	estimated standard deviation of null distribution (usually found with locfdrFit)
null.prop	estimated proportion of statistics that came from the null distribution
vals	vector of all the observed values from the mixture distribution
up	if TRUE, find standard deviation of overexpressed statistics, otherwise find standard deviation of underexpressed statistics

Details

This function is for experienced users or debugging only - all other users should use getParams, which calls this function.

Value

the estimated standard deviation of the over- or underexpressed statistics

Author(s)

Alyssa Frazee

See Also

getParams, find.mean.up, find.mean.down

---

get.numalts *helper function for getParams*

Description

find estimated number of alternative statistics above a set percentile of the estimated null distribution

Usage

```
get.numalts(pctil, null.mean, null.sd, null.prop, vals, up = TRUE)
```

Arguments

pctil	percentile, in (0,1), of the null distribution for which the number of alternative statistics above (if pctil is greater than 0.5) or below (if pctil is less than 0.5) is desired.
null.mean	estimated mean of null distribution (usually found with locfdrFit)
null.sd	estimated standard deviation of null distribution (usually found with locfdrFit)
null.prop	estimated proportion of statistics that came from the null distribution

vals	vector of all the observed values from the mixture distribution
up	if TRUE, get the number of overexpressed statistics above the 100pctil-th percentile of the null distribution, else get the number of underexpressed statistics below the 100pctil-th percentile of the null distribution

Details

This function is for experienced users or debugging only - all other users should use getParams, which calls this function.

Value

a list with elements

num	the estimated number of alternative values above/below val (see val below)
val	the 100pctil-th percentile of the null distribution

Author(s)

Alyssa Frazee

See Also

getParams

---

getAnnotation *download exon information for a given genome*

Description

using the GenomicFeatures package and the UCSC genome browser, creates a data frame of exons (one exon per row) for the specified genome.

Usage

```
getAnnotation(genome, tablename, genes = TRUE, verbose = TRUE)
```

Arguments

genome	Genome (species) for which annotation is desired. A list of supported genomes can be found using rtracklayer:::ucscGenomes()[,"db"]; details on each genome can be seen using rtracklayer:::ucscGenomes().
tablename	UCSC table from which to download exon information. Use supportedTables(genome) to get a list of supported tables for genome.
genes	If TRUE (as it is by default), each exon in the resulting data frame is labeled with the gene it belongs to. Gene information is not available in every table. If FALSE, each exon in the resulting data frame is labeled with the transcript it belongs to.
verbose	If TRUE, updates are printed on screen as annotation download progresses.

Value

A data frame (one row per exon) giving the exon's gene or transcript, location (start/end), and possibly other information.

Note

This function interacts with the online UCSC Genome Browser, so internet connection is required to use this function, and connection speed affects function speed.

Author(s)

Alyssa Frazee

Examples

```
mouse.exons <- getAnnotation("mm9","refGene")
head(mouse.exons)
```

---

getExons *find closest exon(s) to a genomic region*

Description

Given any genomic region (chromosome, start, end), return the closest known exon.

Usage

```
getExons(region, annotation, verbose = TRUE)
```

Arguments

region	length-3 vector (chromosome, start position, end position) of the genomic region of interest. Note that chromosome needs to be in the same format as the chr column of annotation.
annotation	Data frame containing exon information (one row per exon) for the appropriate genome. Must contain columns chr, start, and end. It is recommended that getAnnotation be used to obtain an annotation data frame.
verbose	If TRUE, prints output messages when function finishes.

Value

a list with elements

region	the region argument provided
closestExons	the rows of annotation corresponding to the closest exon to region

Author(s)

Alyssa Frazee

See Also

getAnnotation

Examples

```
## not run:
exons = getAnnotation("hg19","knownGene")
theRegion = c("chr22", 18216902, 18218350)
getExons(theRegion, exons)
foo = getExons(theRegion, exons)
foo
foo$closestExons
```

---

getLimmaInput *fit a linear model to each nucleotide*

Description

Fits the linear model log2(count+0.5) = beta0 + beta1*group + beta2*library.size + [optional confounders] to each nucleotide. From these models, this function constructs an object which can be directly passed to `getTstats` to obtain `limma`'s moderated t statistics for each nucleotide, which we use as a measure of strength of association between group and count (expression). Reads coverage file from a SQLite database (see `makeDb`) and relies heavily on the `limma` package, using `lmFit` as the main workhorse.

Usage

```
getLimmaInput(dbfile, tablename, group, chunksize = 1e+05, adjustvars = NULL, colsubset = NULL)
```

Arguments

`dbfile`	Name/location (as character string) of database (usually ".db") file containing nucleotide by sample coverage.
`tablename`	Name of the table the database contains
`group`	a 0/1 vector grouping the samples (columns) in the database.
`chunksize`	How many rows of the database should be processed at a time?
`adjustvars`	Optional matrix of adjustment variables (e.g. measured confounders, output from SVA, etc.) to use in fitting linear models to each nucleotide.
`colsubset`	Optional vector of column indices of the input file that denote samples you wish to include in analysis. Should NOT include 1 (pos).

Details

It is assumed that the first column in the database is called `pos` and contains genomic position. `group` Must have the one fewer entries than the database denoted by `dbfile` has columns. Larger values of `chunksize` require more memory; smaller values of `chunksize` require more computation time. `adjustvars` must have the same number of rows as `group` has entries. ONLY EXPERIENCED USERS should provide `colsubset`. This option should be used infrequently if at all, reason being that providing `colsubset` will load all of `dbfile` into memory. Mainly used for debugging.

Value

a list with elements

ebobject	A list of five vectors (coefficients, stdev.unscaled, sigma, df.residual, and Amean), mimicking the MArrayLM class in limma. Here, coefficients and stdev.unscaled are only returned for beta1, the coefficient for group, as it is assumed this is the only covariate of interest.
pos	A vector of the same length as those contained in ebobject, giving the genomic positions of each linear model.

Author(s)

Alyssa Frazee

References

Smyth G (2004). "Linear models and empirical Bayes methods for assessing differential expression in microarray experiments." Statistical Applications in Genetics and Molecular Biology 3(1): Article 3.

See Also

getTstats, makeDb, lmFit, MArrayLM-class

Examples

```
## add example here when we have a vignette
```

---

getParams *calculate parameters to use as input for HMM*

Description

Assumes that the moderated t statistics obtained by fitting a linear model to each nucleotide come from a Gaussian mixture distribution, where the four distributions in the mixture represent distributions of t statistics from "underexpressed," "overexpressed," "equally expressed," and "not expressed" nucleotides. getParams estimates the parameters of each of the sub-distributions, as well as the percentage of the mixture distribution each contributes, in order to use these parameters to fit a Hidden Markov Model that classifies the nucleotides.

Usage

```
getParams(tstats, plots = FALSE, plotfile = NULL, verbose = F)
```

Arguments

tstats	Vector containing all moderated t statistics obtained using getTstats.
plots	if TRUE, create diagnostic plots as parameters are estimated
plotfile	Optional string giving a location and PDF file name to which plots should be written, if plots = TRUE. If NULL, plots are created in the available graphics device.
verbose	If TRUE, periodic messages are printed onscreen during estimation.

Details

The standard pipeline here is to feed the output from `getParams` directly into `getRegions` using the "HMM" option.

Value

a list with elements

`params`	list with elements `mean` and `sd`, both 4-item vectors. `mean` gives the respective means of the "not expressed," "equally expressed," "overexpressed," and "underexpressed" distributions; `sd` gives their respective standard deviations.
`stateprobs`	vector of percentages of the mixture distribution that come from the not expressed," "equally expressed," "overexpressed," and "underexpressed" distributions, respectively. It is assumed that "overexpressed" and "underexpressed" t statistics comprise equal percentages of the mixture.

Author(s)

Alyssa Frazee

See Also

`getRegions`

`getParams.failsafe` *helper function for getParams*

Description

When numerical methods `find.mean` and `find.sd` fail, `getParams.failsafe` is used to calculate parameters of the distributions of t statistics originating from over- or underexpressed nucleotides.

Usage

```
getParams.failsafe(null.mean, null.sd)
```

Arguments

`null.mean`	Estimated mean of null distribution (usually from `locfdrFit`)
`null.sd`	Estimated standard deviation of null distribution (usually from `locfdrFit`)

Details

For experienced users/debugging only. Most users should use `getParams` directly.

Value

a list with elements

DEup.mean	estimated mean of overexpressed distribution, calculated as the 95th percentile of the estimated null distribution
DEup.sd	estimated standard deviation of overexpressed distribution, set to be equal to the estimated standard deviation of the null distribution
DEdown.mean	estimated mean of underexpressed distribution, calculated as the 5th percentile of the estimated null distribution
DEdown.sd	estimated standard deviation of underexpressed distribution, set to be equal to the estimated standard deviation of the null distribution

Author(s)

Alyssa Frazee

See Also

getParams

---

getRegions *generate list of regions, classify each as differentially expressed or not*

Description

Using one of three methods, divides the genome (or chromosome) into regions by putting each nucleotide into a state and grouping contiguous nucleotides of the same state into "regions." Regions of states 3 and 4 are "differentially expressed."

Usage

```
getRegions(method, chromosome, pos, tstats, transprobs = c(0.999, 1e-12), stateprobs = NULL, par
```

Arguments

method	Can be one of "HMM" (Hidden Markov Model), "CBS" (circular binary segmentation), or "smoothcut" (t statistics with high enough absolute values are called differentially expressed).
chromosome	Name of chromosome being analyzed - will be printed in output table.
pos	Vector giving genomic positions of the provided t statistics. Must have length equal to that of tstats. pos is returned by getLimmaInput.
tstats	Vector giving moderated t statistics, in proper genomic order.
transprobs	Vector denoting transition probabilities between states, for use in the "HMM" method. Should have length 2, with first element denoting the probability of staying in the same state (should be large), and the second element denoting the probability of moving directly from a differentially expressed state to an equally expressed state or vice versa, or from an overexpressed state to an underexpressed state or vice versa (should be very small). Defaults to c(.999, 1e-12).

stateprobs	Marginal probabilities of being in each of the four hidden states, for use with the "HMM" method. The stateprobs element of getParams generates this.
params	Parameters of the normal distributions representing the four states in the "HMM" method. The params element of getParams generates this.
K	Smoothing parameter used in the "smoothcut" method: t statistics are smoothed using running median; how wide should the window be? Default 25.
tcut	Cutoff used in the "smoothcut" method to classify differential expression: how large in absolute value should a moderated t statistic be in order to be classified as having been generated from a differentially expressed nucleotide? Default 2.
includet	If TRUE, the table in the output will include the average t statistic for each region.
includefchange	If TRUE, the table in the output will include the average estimated fold change (as estimated from the linear models) for each region.
fchange	Required if includefchange = TRUE. Estimated log2 fold changes from the linear models - should have length equal to that of tstats. Usually obtained from the logfchange element of the output of getTstats.

Details

States are labeled numerically in the output as follows: 1="not expressed," 2="equally expressed," 3="overexpressed," 4="underexpressed."

Value

a list with elements

states.norle	data frame with one row per nucleotide, giving its genomic location and predicted hidden state
states	data frame with one row per region, giving its genomic location, length, predicted hidden state, and (if applicable) average t statistic and/or fold change.

Author(s)

Alyssa Frazee

See Also

getTstats, getParams

---

getTstats *calculate moderated t statistic for each nucleotide*

Description

Less-complex version of eBayes in the limma package – only gives output for one covariate of interest. The simplifying was done to conserve memory.

Usage

```
getTstats(fit, trend = FALSE)
```

Arguments

fit	list with elements $sigma, $df.residual, $Amean (if trend=TRUE), $coefficients, and $stdev.unscaled – this can be an lmFit object or an object produced with getLimmaInput, namely, the $ebobject component of a getLimmaInput object.
trend	logical, should an intensity-trend be allowed for the prior variance? (see limma's eBayes). Default is that the prior variance is constant.

Value

list with elements

tt	vector containing moderated t-statistics for each nucleotide
logfchange	vector containing the log base 2 fold change in coverage between the two groups, as estimated by a linear model

Author(s)

eBayes authored by Gordon Smyth, modifications by Alyssa Frazee

References

http://www.bioconductor.org/packages/2.10/bioc/vignettes/limma/inst/doc/usersguide.pdf

See Also

getLimmaInput

---

gsubfn *helper function for read.csv.sql*

Description

see gsubfn in sqldf package - this function is equivalent, but functionality requiring tcltk has been removed.

Usage

```
gsubfn(pattern, replacement, x, backref, USE.NAMES = FALSE, ignore.case = FALSE, env = parent.fr
```

Arguments

pattern	Same as pattern in gsub
replacement	A character string, function, list, formula or proto object. See Details.
x	Same as x in gsub

backref	Number of backreferences to be passed to function. If zero or positive the match is passed as the first argument to the replacement function followed by the indicated number of backreferences as subsequent arguments. If negative then only the that number of backreferences are passed but the match itself is not. If omitted it will be determined automatically, i.e. it will be 0 if there are no backreferences and otherwise it will equal negative the number of back references. It determines this by counting the number of non-escaped left parentheses in the pattern. Also if the function contains an ampersand as an argument then backref will be taken as non-negative and the ampersand argument will get the full match.
USE.NAMES	See USE.NAMES in sapply
ignore.case	If TRUE then case is ignored in the pattern argument.
env	Environment in which to evaluate the replacement function. Normally this is left at its default value.
...	Other gsub arguments

Details

If replacement is a string then it acts like gsub. If replacement is a function then each matched string is passed to the replacement function and the output of that function replaces the matched string in the result. The first argument to the replacement function is the matched string and subsequent arguments are the backreferences, if any. If replacement is a list then the result of the regular expression match is, in turn, matched against the names of that list and the value corresponding to the first name in the list that is match is returned. If there are no names matching then the first unnamed component is returned and if there are no matches then the string to be matched is returned. If backref is not specified or is specified and is positive then the entire match is used to lookup the value in the list whereas if backref is negative then the identified backreference is used. If replacement is a formula instead of a function then a one line function is created whose body is the right hand side of the formula and whose arguments are the left hand side separated by + signs (or any other valid operator). The environment of the function is the environment of the formula. If the arguments are omitted then the free variables found on the right hand side are used in the order encountered. 0 can be used to indicate no arguments. letters, LETTERS and pi are never automatically used as arguments. If replacement is a proto object then it should have a fun method which is like the replacement function except its first argument is the object and the remaining arguments are as in the replacement function and are affected by backref in the same way. gsubfn automatically inserts the named arguments in the call to gsubfn into the proto object and also maintains a count variable which counts matches within strings. The user may optionally specify pre and post methods in the proto object which are fired at the beginning and end of each string (not each match). They each take one argument, the object. Note that if backref is non-negative then internally the pattern will be parenthesized. A utility function cat0 is available. They are like cat and paste except that their default sep value is "".

Value

as in gsub

Author(s)

G. Grothendieck

See Also

strapply

`last` *get the last element*

Description

return last element in a given object (any object with a `tail` method will work).

Usage

```
last(x)
```

Arguments

`x`	any object with a `tail` method

Value

same as `x[length(x)]`

Note

Helper function for `getParams`

Author(s)

Alyssa Frazee

Examples

```
x = c(1:20)
last(x) #returns 20
```

`locfdrFit` *modified version of Efron's locfdr*

Description

Compute local false discovery rates, following the definitions and description in references listed below. Exactly the same as `locfdr`, but returns extra information.

Usage

```
locfdrFit(zz, bre = 120, df = 7, pct = 0, pct0 = 1/4, nulltype = 1, type = 0, plot = 1, mult, ml
```

Arguments

zz	A vector of summary statistics, one for each case under simultaneous consideration. The calculations assume a large number of cases, say length(zz) exceeding 200. Results may be improved by transforming zz so that its elements are theoretically distributed as N(0, 1) under the null hypothesis. See the locfdr vignette for tips on creating zz.
bre	Number of breaks in the discretization of the z-score axis, or a vector of breakpoints fully describing the discretization. If length(zz) is small, such as when the number of cases is less than about 1000, set bre to a number lower than the default of 120. The tornado package keeps this at its default.
df	Degrees of freedom for fitting the estimated density f(z). The tornado package keeps this at its default.
pct	Excluded tail proportions of zz's when fitting f(z). pct=0 includes full range of zz's. pct can also be a 2-vector, describing the fitting range. The tornado package keeps this at its default.
pct0	Proportion of the zz distribution used in fitting the null density f0(z) by central matching. If a 2-vector, e.g. pct0=c(0.25,0.60), the range [pct0[1], pct0[2]] is used. If a scalar, [pct0, 1-pct0] is used. The tornado package keeps this at its default.
nulltype	Type of null hypothesis assumed in estimating f0(z), for use in the fdr calculations. 0 is the theoretical null N(0; 1), 1 is maximum likelihood estimation, 2 is central matching estimation, 3 is a split normal version of 2. The tornado package fixes this at 1.
type	Type of fitting used for f; 0 is a natural spline, 1 is a polynomial, in either case with degrees of freedom df [so total degrees of freedom including the intercept is df+1.] The tornado package fixes this at 0.
plot	Plots desired. 0 gives no plots. 1 gives single plot showing the histogram of zz and fitted densities f and p0 f0. 2 also gives plot of fdr, and the right and left tail area Fdr curves. 3 gives instead the f1 cdf of the estimated fdr curve; plot=4 gives all three plots. The tornado package allows choices 0 and 1; equivalent to plots = F and plots = T in getParams.
mult	Optional scalar multiple (or vector of multiples) of the sample size for calculation of the corresponding hypothetical Efdr value(s). This is not used in the tornado package.
mlests	Optional vector of initial values for (delta0, sigma0) in the maximum likelihood iteration. The tornado package includes these values in an updated run of locfdrFit if they are suggested via warning in the first run.
main	Main heading for the histogram plot when plot>0.
sw	Determines the type of output desired. 2 gives a list consisting of the last 5 values listed under Value below. 3 gives the square matrix of dimension bre-1 representing the influence function of log(fdr). Any other value of sw returns a list consisting of the first 5 (6 if mult is supplied) values listed below. The tornado package fixes this at 0.
verbose	if TRUE, various messages are printed onscreen during getParams.

Details

Generally not used directly in the tornado package, but is a workhorse for getParams.

Value

See the locfdr manual for returned values. locfdrFit returns the following additional elements:

yt	bin heights
mlest.lo	if a warning that mlest should be used in a re-run of locfdrFit, the suggested first element of mlest.
mlest.hi	if a warning that mlest should be used in a re-run of locfdrFit, the suggested second element of mlest.
needsfix	0 if no warning to fix the run with mlest parameters, 1 otherwise
nulldens	a rough estimate of y-axis values for f0(x)
fulldens	a rough estimate of y-axis values for f(x)

Author(s)

Bradley Efron, slight modifications by Alyssa Frazee

References

http://cran.r-project.org/web/packages/locfdr/locfdr.pdf

See Also

getParams

---

makeDb *create SQLite database from text file*

Description

Dumps the contents of a table (saved as a text file) into a SQLite database, performing some filtering along the way.

Usage

```
makeDb(dbfile, textfile, tablename, sep = "\t", cutoff = 5)
```

Arguments

dbfile	Character string giving the file name/location of the database to be created. Generally ends in .db.
textfile	The text file containing the table to be dumped into dbfile.
tablename	Character string containing name to give the table inside dbfile.
sep	The separator used in textfile. The tornado pipeline creates tab-separated text files, so "\t" is the default.
cutoff	Rows in textfile must have at least one entry (not counting the first column, which is assumed to hold genomic position) greater than cutoff to be included in dbfile.

Value

No return, but writes the file `dbfile` containing table `tablename` by filtering `textfile` according to `cutoff`.

Note

The workhorse of this function is a modified version of `read.csv.sql`, found in the `sqldf` package.

Author(s)

Alyssa Frazee

`match.funfn` *Generic extended version of R match.fun*

Description

A generic `match.fun`.

Usage

```
match.funfn(FUN, descend = TRUE)
```

Arguments

`FUN`	Function, character name of function or formula describing function.
`descend`	logical; control whether to search past non-function objects.

Details

The default method is the same as `match.fun` and the `formula` method is the same as `as.function.formula`. This function can be used within the body of a function to convert a function specification whether its a function, character string or formula into an actual function.

Value

Returns a function.

Note

This is exactly the same as `match.funfn` in the `gsubfn` package. Do not load the `gsubfn` package to use this function, as `gsubfn` loads `tcltk`, which is not advised.

Author(s)

G. Grothendieck

ostrapply *helper function for read.csv.sql*

Description

Same as the strapply or ostrapply functions in gsubfn package, but with tcltk capabilities removed.

Usage

```
ostrapply(X, pattern, FUN = function(x, ...) x, ignore.case = FALSE, ..., empty = NULL, simplify
```

Arguments

X	list or (atomic) vector of character strings to be used
pattern	character string containing a regular (or character string for fixed=TRUE to be matched in the given character vector.
FUN	a function, formula, character string, list or proto object to be applied to each element of X. See discussion in gsubfn.
ignore.case	If TRUE then case is ignored in the pattern argument.
...	optional arguments to gsubfn.
empty	If there is no match to a string return this value.
simplify	logical or function. If logical, should the result be simplified to a vector or matrix, as in sapply if possible? If function, that function is applied to the result with each component of the result passed as a separate argument. Typically if the form is used it will typically be specified as rbind.
USE.NAMES	logical; if TRUE and if X is character, use X as 'names' for the result unless it had names already.
combine	combine is a function applied to the components of the result of FUN. The default is "c". "list" is another common choice. The default may change to be "list" in the future.

Details

See strapply in gsubfn. In the tornado package this function should rarely be called on its own. It is an internal process of read.csv.sql, which is an internal process of makeDb.

Value

A list of character strings.

Author(s)

G. Grothendieck

References

http://cran.r-project.org/web/packages/gsubfn/gsubfn.pdf

See Also

makeDb,link{read.csv.sql}

---

plotExon *plot pipeline data/results for a given exon*

Description

creates a 3-paneled plot of a selected exon: panel 1 = genomic position vs. raw coverage data, panel 2 = genomic position vs. moderated t statistic from linear model at that position, panel 3 = genomic position vs. predicted state for that position, with annotated exons overlaid.

Usage

```
plotExon(getRegionObject, ind = NULL, exonname = NULL, tstats, pos, annotation, counts, group, b
```

Arguments

getRegionObject	The name of an object created with getRegions.
ind	index in the provided annotation of the exon you wish to plot.
exonname	name of the exon (as listed in the provided annotation) you wish to plot
tstats	Vector of t-statistics that was used to create getRegionObject.
pos	vector of genomic positions corresponding to tstats
annotation	data frame containing exon annotation to use (see getAnnotation). Must contain a "name" column listing the exon names. (The column exon_id in a getAnnotation object can be re-named to name).
counts	Raw coverage data used to obtain tstats. This can be provided in one of three forms: (1) a string indicating the location/file name of a SQLite database containing the counts, usually created with makeDb; (2) a string indicating the location/file name of a text file containing coverage – this will get loaded into memory!! – or (3) an already-loaded matrix containing the raw data. Note that counts must have the same number of rows as pos and tstats have elements, and the rows must correspond to genomic position pos.
group	a vector containing the group labels for the columns of counts. Only 2 groups are permitted at this time. These labels are used in the plot's legend, so generally they are character strings (rather than, say, 0/1).
bppad	the number of bases to plot outside of the designated region (default 50). Essentially, use this to "zoom" in (decrease bppad) or out (increase bppad) on the plotted region.
axpad	how much wider (in bases) you'd like the x-axis to be, compared to the plotted area
prettyskips	If TRUE, plot counts/states/t-statistics contiguously, even if there are zero entries between them (i.e., even though pos may not indicate that contiguous postions are being plotted). Note that in general, when plotting just one exon, this is not an issue as exons tend to contain contiguous data. So, prettyskips will only affect areas outside the exon, i.e., prettyskips has larger impact if bppad is large. Also, note that prettyskips = FALSE is not allowed at this time.

skiplines	if TRUE, add a light vertical line to the plot indicating an eliminated "low-coverage" nucleotide
countsheader	If TRUE, the counts matrix contains a header row. Not usually the case if counts is a database or already-loaded matrix.
countssep	If reading counts from a text file, the separator used in that file.
tabname	If counts is a database file, the name of the table that was dumped into that database. (See tablename in makeDb)
plotfile	Optional string containing a file you'd like to put the plot into (if NULL, plot appears interactively). Should have a .jpg extension.
width	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 900.
height	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 750.
plottitle	Optional main title to use on your plot. Defaults to chromosome: start-end (referring to plotted REGION)
chromosome	The chromosome corresponding to the exon being plotted, in the same format as chromosomes are listed in the supplied annotation.
legendloc	Can be one of "topright","bottomright","topleft",or "bottomleft" indicating where the legend (indicating group label on raw count plot) should be located. Defaults to "bottomleft."

Value

Plots (either on screen or in the supplied .jpg file) the following: top panel = genomic position vs. raw coverage data, middle panel = genomic position vs. moderated t statistic from linear model at that position, bottom panel = genomic position vs. predicted state for that position, with annotated exons overlaid. States are as follows: gray = not expressed, black = equally expressed, red = overexpressed (in whatever group = 1 represented in getRegions), green = underexpressed.

Note

Provide exactly one of ind and exonname.

Author(s)

Alyssa Frazee

See Also

getRegions, makeDb

plotGene *plot pipeline data/results for a given gene*

Description

creates a 3-paneled plot of a selected gene: panel 1 = genomic position vs. raw coverage data, panel 2 = genomic position vs. moderated t statistic from linear model at that position, panel 3 = genomic position vs. predicted state for that position, with annotated exons overlaid.

Usage

```
plotGene(getRegionObject, ind = NULL, genename = NULL, tstats, pos, annotation, counts, group, b
```

Arguments

getRegionObject	The name of an object created with getRegions.
ind	index in the provided annotation of the exon you wish to plot.
genename	name of the gene (as listed in the provided annotation) you wish to plot
tstats	Vector of t-statistics that was used to create getRegionObject.
pos	vector of genomic positions corresponding to tstats
annotation	data frame containing exon annotation to use (see getAnnotation). Must contain a "geneName" column listing the exon names. (The column gene in a getAnnotation object can be re-named to geneName).
counts	Raw coverage data used to obtain tstats. This can be provided in one of three forms: (1) a string indicating the location/file name of a SQLite database containing the counts, usually created with makeDb; (2) a string indicating the location/file name of a text file containing coverage – this will get loaded into memory!! – or (3) an already-loaded matrix containing the raw data. Note that counts must have the same number of rows as pos and tstats have elements, and the rows must correspond to genomic position pos.
group	a vector containing the group labels for the columns of counts. Only 2 groups are permitted at this time. These labels are used in the plot's legend, so generally they are character strings (rather than, say, 0/1).
bppad	the number of bases to plot outside of the designated region (default 50). Essentially, use this to "zoom" in (decrease bppad) or out (increase bppad) on the plotted region.
axpad	how much wider (in bases) you'd like the x-axis to be, compared to the plotted area
prettyskips	If TRUE, plot counts/states/t-statistics contiguously, even if there are zero entries between them (i.e., even though pos may not indicate that contiguous postions are being plotted). Note that in general, when plotting just one exon, this is not an issue as exons tend to contain contiguous data. So, prettyskips will only affect areas outside the exon, i.e., prettyskips has larger impact if bppad is large. Also, note that prettyskips = FALSE is not allowed at this time.
skiplines	if TRUE, add a light vertical line to the plot indicating an eliminated "low-coverage" nucleotide

countsheader	If TRUE, the counts matrix contains a header row. Not usually the case if counts is a database or already-loaded matrix.
countssep	If reading counts from a text file, the separator used in that file.
tabname	If counts is a database file, the name of the table that was dumped into that database. (See tablename in makeDb)
plotfile	Optional string containing a file you'd like to put the plot into (if NULL, plot appears interactively). Should have a .jpg extension.
width	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 900.
height	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 750.
plottitle	Optional main title to use on your plot. Defaults to chromosome: start-end (referring to plotted REGION)
chromosome	The chromosome corresponding to the exon being plotted, in the same format as chromosomes are listed in the supplied annotation.
legendloc	Can be one of "topright","bottomright","topleft",or "bottomleft" indicating where the legend (indicating group label on raw count plot) should be located. Defaults to "bottomleft."

Value

Plots (either on screen or in the supplied .jpg file) the following: top panel = genomic position vs. raw coverage data, middle panel = genomic position vs. moderated t statistic from linear model at that position, bottom panel = genomic position vs. predicted state for that position, with annotated exons overlaid. States are as follows: gray = not expressed, black = equally expressed, red = overexpressed (in whatever group = 1 represented in getRegions), green = underexpressed.

Note

Provide exactly one of ind and genename. Recommendation is to provide geneName.

Author(s)

Alyssa Frazee

See Also

getRegions, makeDb

---

plotRegion *plot pipeline data/results for a given region*

Description

creates a 3-paneled plot of a selected region: panel 1 = genomic position vs. raw coverage data, panel 2 = genomic position vs. moderated t statistic from linear model at that position, panel 3 = genomic position vs. predicted state for that position, with annotated exons overlaid.

Usage

```
plotRegion(getRegionObject, ind, tstats, pos, annotation, counts, group, bppad = 50, axpad = 50,
```

Arguments

getRegionObject	The name of an object created with getRegions.
ind	index in the provided getRegionObject$states of the region you wish to plot.
tstats	Vector of t-statistics that was used to create getRegionObject.
pos	vector of genomic positions corresponding to tstats
annotation	data frame containing exon annotation to use (see getAnnotation). Must contain a "name" column listing the exon names. (The column exon_id in a getAnnotation object can be re-named to name).
counts	Raw coverage data used to obtain tstats. This can be provided in one of three forms: (1) a string indicating the location/file name of a SQLite database containing the counts, usually created with makeDb; (2) a string indicating the location/file name of a text file containing coverage – this will get loaded into memory!! – or (3) an already-loaded matrix containing the raw data. Note that counts must have the same number of rows as pos and tstats have elements, and the rows must correspond to genomic position pos.
group	a vector containing the group labels for the columns of counts. Only 2 groups are permitted at this time. These labels are used in the plot's legend, so generally they are character strings (rather than, say, 0/1).
bppad	the number of bases to plot outside of the designated region (default 50). Essentially, use this to "zoom" in (decrease bppad) or out (increase bppad) on the plotted region.
axpad	how much wider (in bases) you'd like the x-axis to be, compared to the plotted area
prettyskips	If TRUE, plot counts/states/t-statistics contiguously, even if there are zero entries between them (i.e., even though pos may not indicate that contiguous postions are being plotted). Note that in general, when plotting just one exon, this is not an issue as exons tend to contain contiguous data. So, prettyskips will only affect areas outside the exon, i.e., prettyskips has larger impact if bppad is large. Also, note that prettyskips = FALSE is not allowed at this time.
skiplines	if TRUE, add a light vertical line to the plot indicating an eliminated "low-coverage" nucleotide
countsheader	If TRUE, the counts matrix contains a header row. Not usually the case if counts is a database or already-loaded matrix.
countssep	If reading counts from a text file, the separator used in that file.
tabname	If counts is a database file, the name of the table that was dumped into that database. (See tablename in makeDb)
plotfile	Optional string containing a file you'd like to put the plot into (if NULL, plot appears interactively). Should have a .jpg extension.
width	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 900.
height	Only used when plotfile is non-null: width (in pixels) of resulting jpg. Defaults to 750.

plottitle	Optional main title to use on your plot. Defaults to chromosome: start-end (referring to plotted REGION)
chromosome	The chromosome corresponding to the exon being plotted, in the same format as chromosomes are listed in the supplied annotation.
legendloc	Can be one of "topright","bottomright","topleft",or "bottomleft" indicating where the legend (indicating group label on raw count plot) should be located. Defaults to "bottomleft."

Value

Plots (either on screen or in the supplied .jpg file) the following: top panel = genomic position vs. raw coverage data, middle panel = genomic position vs. moderated t statistic from linear model at that position, bottom panel = genomic position vs. predicted state for that position, with annotated exons overlaid. States are as follows: gray = not expressed, black = equally expressed, red = overexpressed (in whatever group = 1 represented in getRegions), green = underexpressed.

Author(s)

Alyssa Frazee

See Also

getRegions, makeDb

---

read.csv.sql *main workhorse of makeDb*

Description

Edited version of read.csv.sql in the sqldf package (tcltk functionality removed), used mainly to create a SQLite database from a text file. The function makeDb wraps this function in the tornado package, so no need to execute this directly.

Usage

```
read.csv.sql(file, sql = "select * from file", header = TRUE, sep = ",", row.names, eol, skip, f
```

Arguments

file	A file path or a URL (beginning with http:// or ftp://). If the filter argument is used and no file is to be input to the filter then file can be omitted, NULL, NA or "". The textfile argument from makeDb is used here.
sql	character string holding an SQL statement. The table representing the file should be referred to as file.
header	as in read.csv
sep	as in read.csv
row.names	as in read.csv
eol	Character that ends lines
skip	Skip indicated number of lines in input file.

filter	see read.csv.sql in sqldf
nrows	Number of rows used to determine column types. It defaults to 50. Using -1 causes it to use all rows for determining column types. This argument is rarely needed.
field.types	A list whose names are the column names and whose contents are the SQLite types (not the R class names) of the columns. Specifying these types improves how fast it takes. Unless speed is very important this argument is not normally used.
comment.char	If specified this character and anything following it on any line of the input will be ignored.
dbname	As in sqldf except that the default is tempfile(). Specifying NULL will put the database in memory which may improve speed but will limit the size of the database by the available memory. When using tornado, tempfile() is not used: dbname must be provided as an argument to makeDb.
drv	ignored: the only drive used can be SQLite.
...	arguments passed to sqldf.

Details

This function is entirely wrapped by makeDb – experienced users may use this for debugging, but otherwise not usually necessary to call directly.

Value

If the sql statement is a select statement then a data frame is returned. In tornado, this is never the case.

References

http://cran.r-project.org/web/packages/sqldf/sqldf.pdf

See Also

makeDb

---

sqldf *helper function for read.csv.sql*

Description

used internally by read.csv.sql, which drives the makeDb function. Not necessary to call this function directly when using the tornado package.

Usage

```
sqldf(x, stringsAsFactors = FALSE, row.names = FALSE, envir = parent.frame(), method = getOption
```

Details

For arguments, value, and other information, see sqldf - this function is a direct copy of that function.

References

http://cran.r-project.org/web/packages/sqldf/sqldf.pdf

See Also

makeDb

---

strapply *helper function for read.csv.sql*

Description

The same as strapply in the gsubfn package, but with tcltk capabilities removed.

Usage

```
strapply(X, pattern, FUN = function(x, ...) x, backref = NULL, ..., empty = NULL, ignore.case =
```

Arguments

X	list or (atomic) vector of character strings to be used
pattern	character string containing a regular (or character string for fixed=TRUE to be matched in the given character vector.
FUN	a function, formula, character string, list or proto object to be applied to each element of X. See discussion in gsubfn.
backref	see gsubfn
...	optional arguments to gsubfn
empty	If there is no match to a string return this value.
ignore.case	If TRUE then case is ignored in the pattern argument.
perl	If TRUE then engine="R" is used with perl regular expressions. It is required to keep this argument at TRUE, since tcl engine capabilities have been removed from this function.
engine	Should always be set to "R", since the tcl engine is not available in the tornado package.
simplify	logical or function. If logical, should the result be simplified to a vector or matrix, as in sapply if possible? If function, that function is applied to the result with each component of the result passed as a separate argument. Typically if the form is used it will typically be specified as rbind.
USE.NAMES	logical; if TRUE and if X is character, use X as 'names'w for the result unless it had names already.
combine	combine is a function applied to the components of the result of FUN. The default is "c". "list" is another common choice. The default may change to be "list" in the future.

Details

See details in gsubfn package.

Value

A list of character strings

Note

Does not need to be used directly in tornado; makeDb wraps this entirely.

Author(s)

G. Grothendieck

References

http://cran.r-project.org/web/packages/gsubfn/gsubfn.pdf

See Also

makeDb

---

supportedTables *print list of supported (downloadable) tables for a given genome*

Description

To be used with getAnnotation: provides a list of available tables from UCSC for any given genome.

Usage

```
supportedTables(genome)
```

Arguments

genome	Character string giving the genome of interest. Available genomes can be seen with rtracklayer:::ucscGenomes()[,"db"].

Value

prints a list of available tables for genome.

Author(s)

Alyssa Frazee

See Also

getAnnotation

Examples

```
supportedTables("mm10")
mouse.exons = getAnnotation("mm10","refGene") #refGene appears in printed output of supportedTables("mm10")
```

Index