Package 'goldmine'

September 25, 2014

Maintainer Jeffrey Bhasin <jeffb@case.edu>

Author Jeffrey Bhasin <jeffb@case.edu>

Version 0.1

License GPL-3

Title goldmine: Genomic context annotation for any set of genomic ranges using UCSC Genome Browser tables

Description Goldmine obtains data by direct downloading and updating of a local mirror of select UCSC Genome Browser annotation tables. The R package contains functions to assess genomic context of any given set of genomic ranges by performing overlaps with regions of annotated genomic features and produce long, short, and plot outputs.

Depends GenomicRanges, data.table, stringr, ggplot2, parallel, IRanges

Imports httr, RCurl, R.utils, gtools, Matching, rms, grid, gridExtra,RColorBrewer, reshape

R topics documented:

goldmine-package . . . 2
addGenes . . . 2
addNearest . . . 3
calcEnrichmentBinom . . . 3
calcMotifCounts . . . 4
drawBackgroundSetPropensity . . . 4
getCpgFeatures . . . 5
getFeatures . . . 5
getGenes . . . 6
getSeqMeta . . . 6
getUCSCTable . . . 7
ggnice . . . 8
gmWrite . . . 8
goldmine . . . 8
makeDT . . . 9
makeGRanges . . . 10
plotCovarDistance . . . 10
plotCovarHistograms . . . 11
plotCovarHistogramsOverlap . . . 11

plotCovarQQ 12
readFIMO 12
readUCSCAnnotation 12
runFIMO 13
sortDT 13
sortGRanges 13
testEnrichment 14
writeBEDFromGRanges 14

Index **15**

`goldmine-package` *goldmine: Genomic context annotation for any set of genomic ranges using UCSC Genome Browser tables*

Description

goldmine: Genomic context annotation for any set of genomic ranges using UCSC Genome Browser tables

---

`addGenes` *Add columns with distance to nearest gene and gene symbol(s)*

Description

Add columns with distance to nearest gene and gene symbol(s)

Usage

```
addGenes(query, geneset, genome, cachedir)
```

Arguments

`query`	Genomic regions to find nearest genes for as a GRanges, data.frame, or data.table with the columns "chr", "start", and "end"
`geneset`	Select one of "ucsc" for the UCSC Genes (from the knownGene table), "refseq" for RefSeq genes (from the refFlat table), or "ensembl" for the Ensembl genes (from the ensGene table)
`genome`	UCSC genome name to use (e.g. hg19, mm10)
`cachedir`	Path where cached UCSC tables are stores

addNearest — *Add columns to query with distance to nearest subject and subject id(s)*

Description

Add columns to query with distance to nearest subject and subject id(s)

Usage

```
addNearest(query, subject, id = "name", prefix = "subject")
```

Arguments

query	Genomic regions to find nearest genes for as a GRanges, data.frame, or data.table with the columns "chr", "start", and "end"
id	Column name of the id field in subject to report as the nearest id(s). In case of ties, a comma separated list will be returned.
prefix	Append this string to names of the added columns
query	Genomic regions to find nearest genes for as a GRanges, data.frame, or data.table with the columns "chr", "start", and "end"

calcEnrichmentBinom — *Perform binomial test of enrichment for motifs using counts of occurrences in two sequence sets*

Description

The *.counts matrix objects must first be generated using `calcMotifCounts`. The current implementation only considers if a sequence has at least one occurrence of the motif or not, and does not account for or weight multiple occurrences of a motif in a single sequence. The contingency table is simply based on the number of sequences which contain at least one occurrence of each motif.

Usage

```
calcEnrichmentBinom(seq1.counts, seq1.nSeqs, seq2.counts, seq2.nSeqs)
```

Arguments

seq1.counts	output object (matrix) from `calcMotifCounts` for first sequence set
seq1.nSeqs	number of sequences in first sequence set
seq2.counts	output object from `calcMotifCounts` for second sequence set
seq2.nSeqs	number of sequences in second sequence set

Value

dataframe of output results including p-values (unadjusted)

calcMotifCounts *Generate counts of motif occurrences in each sequence*

Description

Generate counts of motif occurrences in each sequence

Usage

```
calcMotifCounts(fimo.out, q.cutoff)
```

Arguments

fimo.out	dataframe from readFIMO dataframe object
q.cutoff	only count a motif if the q-value is less than this cutoff

Value

matrix with pairwise counts of each motif and each sequence

drawBackgroundSetPropensity *Draw a matched reference set from a reference pool*

Description

Use propensity score matching to create a covariate-matched reference set. Note that the matching function is sensitive to the starting order of the input data. This order is required as a variable so it can be fixed between runs.

Usage

```
drawBackgroundSetPropensity(target.seq, target.meta, pool.seq, pool.meta,
  formula, start.order, n = 1)
```

Arguments

target.seq	DNAStringSet object of the target set
target.meta	data.frame object with the covariates of the target set
pool.seq	DNAStringSet object of the reference pool to draw covariate matched reference set from
pool.meta	data.frame object with the covariates
formula	an as.formula object for the regression used to generate propensity scores
start.order	vector of starting order for matching (must be a sequence of integers in any order from 1 to the total number of sequences in both target.seq and pool.seq)

Value

DNAStringSet object of a covariate-matched reference set

getCpgFeatures *Generate feature sets based on CpG island, shore, and shelf regions*

Description

Uses the "cpgIslandExt" table to generate shore (+/- 2kb from islands) and shelf (+/- 2kb from shores) regions.

Usage

```
getCpgFeatures(genome, cachedir)
```

Arguments

genome	See goldmine()
cachedir	See goldmine()

getFeatures *Obtain feature sets from UCSC genome browser tables*

Description

Given a vector of table names from the UCSC genome browser that all contain "chrom", "chromStart", and "chromEnd" fields, converts them to input suitable for the goldmine() "features" argument.

Usage

```
getFeatures(tables = c("wgEncodeRegDnaseClusteredV2",
  "wgEncodeRegTfbsClusteredV3", "tfbsConsSites", "cosmic", "oreganno",
  "vistaEnhancers", "phastConsElements100way"), genome, cachedir)
```

Arguments

tables	A vector of table names from UCSC (default: set of useful tables).
genome	See goldmine()
cachedir	See goldmine()

getGenes *Load table of gene ranges via UCSC Genome Browser tables*

Description

Load table of gene ranges via UCSC Genome Browser tables

Usage

```
getGenes(geneset = "ucsc", genome, cachedir = NULL)
```

Arguments

geneset	Select one of "ucsc" for the UCSC Genes (from the knownGene table), "refseq" for RefSeq genes (from the refFlat table), or "ensembl" for the Ensembl genes (from the ensGene table)
genome	UCSC genome name to use (e.g. hg19, mm10)
cachedir	Path where cached UCSC tables are stores

getSeqMeta *Calculate covariates for each sequence in a DNAStringSet*

Description

Calculate covariates for each sequence in a DNAStringSet

Usage

```
getSeqMeta(ranges, bsgenome, genome, cachedir)
```

Arguments

myseq	DNAStringSet object of the sequence set

Value

dataframe with standard covariates added

`getUCSCTable` *Load an annotation table from the UCSC Genome Browser as an R data.frame*

Description

If only `table` and `genome` are given, the function will load the data directly into the R workspace. If `cachedir` is a path to a directory, this directory will be used to maintain a cachedir cache of UCSC tables so they do not need to be re-downloaded on each call. If the data already exists and `sync=TRUE`, the function will only re-download and re-extract if the modified dates are different between the cachedir and remote copies.

Usage

```
getUCSCTable(table, genome, cachedir = NULL, version = "latest",
  sync = TRUE, url = "http://hgdownload.cse.ucsc.edu/goldenPath/",
  fread = TRUE)
```

Arguments

`table`	The UCSC string specific for the table to sync (e.g. "knownGene", "kgXref", etc)
`genome`	The UCSC string specific to the genome to be downloaded (e.g. "hg19", "hg19", "mm10", etc)
`cachedir`	A path to a directory where a cachedir cache of UCSC tables are stored. If equal to `NULL` (default), the data will be downloaded to temporary files and loaded on the fly.
`version`	If "latest" (default) then use the newest version of the table available. If set to a timestamp string of an archived table (format: YYYY-MM-DD-HH-MM-SS), then load this specific version. Obtain these strings by examining the file names under your cache directory. An archive file with a date stamp is saved automatically with each download of a new version. This feature only works if you have a cachedir cache that contains the desired versions.
`sync`	If `TRUE`, then check if a newer version is available and download if it is. If `FALSE`, skip this check. Only has an effect if a cachedir cache directory (`cachedir`) is given.
`url`	The root of the remote http URL to download UCSC data from (set by default to `http://hgdownload.cse.ucsc.edu/goldenPath/`)

Value

A data.frame of the desired UCSC table.

ggnice — *Clean ggplot2 theme*

Description

Remove gridlines from ggplot2.

Usage

```
ggnice()
```

gmWrite — *Write individual CSV files to disk from the output of goldmine()*

Description

Write a CSV file for each output table in a goldmine() output list object.

Usage

```
gmWrite(gm, path = ".")
```

Arguments

gm	The output list object from goldmine().
path	The directory to write the files into (default: current working directory).

goldmine — *Explore relationships between a set of genomic ranges with genes and features*

Description

Computes the overlap between a query set of genomic ranges given as a GenomicRanges, data.frame, or data.table with gene and feature sets of interest. Reports both summarized overlaps (same number of rows as the query - a "wide format") and in separate tables, individual overlap events (one row for each pair of overlapping query and gene/feature item - a "long format" similar to an inner join).

Usage

```
goldmine(query, genes = getGenes(geneset = "ucsc", genome = genome, cachedir =
  cachedir), features = getFeatures(genome = genome, cachedir = cachedir),
  genome, cachedir)
```

Arguments

query	A GenomicRanges, data.frame, or data.table of regions to annotate. If a data.frame or data.table, must contain the columns "chr", "start", "end", where the "start" coordinates are 1-based. All additional columns will be retained in the output object.
genes	Genes of interest from the output table of getGenes().
features	A list() of GenomicRanges, data.table, or data.frame objects giving feature sets of interest.
genome	The UCSC name specific to the genome of the query coordinates (e.g. "hg19", "hg18", "mm10", etc)
cachedir	A path to a directory where a local cache of UCSC tables are stored. If equal to NULL (default), the data will be downloaded to temporary files and loaded on the fly.

Value

A list: "context" shows a percent overlap for each range in the query set with gene model regions and each feature set, "genes" shows a detailed view of each query region overlap with individual gene isoforms, "features" is a list of tables which for each given feature contain a row for each instance of a query region overlapping with a feature region.

---

makeDT *Make a data.table from a GRanges or a data.frame*

Description

Given a data.frame or GRanges, a data.table object will be created. If the input is already a data.table, it is simply returned.

Usage

```
makeDT(obj)
```

Arguments

obj	A data.frame or GRanges

Value

A data.table made from the data in obj.

makeGRanges — *Make a GRanges from a data.frame or data.table with the fields "chr", "start", and "end"*

Description

Given a data.frame or data.table with the columns "chr", "start", and "end", a GenomicRanges (GRanges) object will be created. All other columns will be passed on as metadata. If the input is already a GRanges, it is simply returned. If the column "strand" exists, it will be set as the strand.

Usage

```
makeGRanges(obj, strand = F)
```

Arguments

obj	A data.frame or data.table with columns "chr", "start", and "end" and any other columns
strand	Use the information in the "strand" column to set strand in the GRanges, if it is present.

Value

A GRanges made from the data in obj.

plotCovarDistance — *Plot horizontal graph of covariate distance from different propensity models*

Description

Horizontal graph plots a point for each variable that represents the distance between that variable's value in orig.meta and each of the dataframes in list.meta

Usage

```
plotCovarDistance(orig.meta, list.meta, cols)
```

Arguments

orig.meta	dataframe of covariates from the target set you are trying to match
list.meta	a list of dataframes of covariates from other sets you want to compare to the target set
cols	vector of which columns to use from the dataframes above

Value

plots to active graphics device

plotCovarHistograms

Plot histograms in a grid for arbitrary number of variables

Description

Plots non-overlapping single histograms in a grid for all covariate data in a dataframe.

Usage

```
plotCovarHistograms(seq.meta, cols)
```

Arguments

seq.meta	data.frame of sequence covariates
cols	vector of which columns to plot histograms for from seq.meta

Value

plot sent to current graphics device

plotCovarHistogramsOverlap

Plot overlapping histograms for any number of variables from 2 sets

Description

Plots a grid of overlapping histograms. Data for seq1 will appear in red and seq2 in blue. The region where the distributions will appear in purple.

Usage

```
plotCovarHistogramsOverlap(seq1.meta, seq2.meta, cols, plot.ncols = 3,
  main = "")
```

Arguments

seq1.meta	dataframe of covariates from first distribution
seq2.meta	dataframe of covariates from second distribution
cols	which columns in the *.meta dataframes contain covariate data to plot
plot.ncols	number of columns in the plotted grid
main	title for the grid of plots (useful if you want to put on the formula you used to generate seq2.meta)

Value

plot to active graphics device

plotCovarQQ *Plot QQ plots in a grid for arbitrary number of variables*

Description

Creates QQ plots to compare two distributions for any number of variables

Usage

```
plotCovarQQ(orig.meta, list.meta, cols, plot.ncols = 3)
```

Arguments

orig.meta	data frame from the original distribution
list.meta	list of data frames for all the distributions to compare to for each variable
cols	which columns have covariates to plot from the above dataframes
plot.ncols	how many columns the plotted grid should have

Value

plot to active graphics device

readFIMO *Read in a FIMO output file*

Description

Wrapper of read.table() with correct options for reading in a FIMO output text file.

Usage

```
readFIMO(fimo.out.path)
```

Arguments

fimo.out.path	path to FIMO output text file

readUCSCAnnotation *Read UCSC table files from disk and join all related tables*

Description

Read UCSC table files from disk and join all related tables

Usage

```
readUCSCAnnotation(genome = "hg19", path = "")
```

`runFIMO` *Wrapper to call FIMO using system()*

Description

If FIMO from the MEME Suite is installed and in the current PATH, this provides an easy interface to run it from R.

Usage

```
runFIMO(out.path, fasta.path, motifs.path)
```

Arguments

`out.path`	path where output will be written
`fasta.path`	path to FASTA file of input sequences
`motifs.path`	path to MEME format file containing motif database to use

`sortDT` *Sort a data.frame, data.table, or GRanges by chr (accounting for mixed string and numeric names), start, end and return a data.table*

Description

Sort a data.frame, data.table, or GRanges by chr (accounting for mixed string and numeric names), start, end and return a data.table

Usage

```
sortDT(obj)
```

Arguments

`obj`	A data.frame, data.table, or GRanges

`sortGRanges` *Sort a data.frame, data.table, or GRanges by chr (accounting for mixed string and numeric names), start, end and return a GRanges*

Description

Sort a data.frame, data.table, or GRanges by chr (accounting for mixed string and numeric names), start, end and return a GRanges

Usage

```
sortGRanges(obj)
```

Arguments

`obj`	A data.frame, data.table, or GRanges

`testEnrichment` *Make a GRanges from a data.frame or data.table with the fields "chr", "start", and "end"*

Description

Given a data.frame or data.table with the columns "chr", "start", and "end", a GenomicRanges (GRanges) object will be created. All other columns will be passed on as metadata. If the input is already a GRanges, it is simply returned.

Usage

```
testEnrichment(query, background, features)
```

Arguments

`obj`	A data.frame or data.table with columns "chr", "start", and "end" and any other columns

Value

A GRanges made from the data in obj.

`writeBEDFromGRanges` *Write a BED format file from a GenomicRanges object*

Description

Creates BED file suitable for upload as a custom track to the UCSC genome browser. Note that start coordinates are 0-based in the BED format.

Usage

```
writeBEDFromGRanges(gr, file, name = NULL)
```

Arguments

`gr`	A GenomicRanges object.
`file`	Filename of the BED file to write.
`name`	Column name to use for the name field in the BED file (optional)

Index

addGenes, 2
addNearest, 3

calcEnrichmentBinom, 3
calcMotifCounts, *3*, 4

drawBackgroundSetPropensity, 4

getCpgFeatures, 5
getFeatures, 5
getGenes, 6
getSeqMeta, 6
getUCSCTable, 7
ggnice, 8
gmWrite, 8
goldmine, 8
goldmine-package, 2

makeDT, 9
makeGRanges, 10

plotCovarDistance, 10
plotCovarHistograms, 11
plotCovarHistogramsOverlap, 11
plotCovarQQ, 12

readFIMO, *4*, 12
readUCSCAnnotation, 12
runFIMO, 13

sortDT, 13
sortGRanges, 13

testEnrichment, 14

writeBEDFromGRanges, 14