Package 'chipenrich'

April 14, 2017

Type Package

Title Gene Set Enrichment For ChIP-seq Peak Data

Version 1.12.1

Date 2016-03-09

Description ChIP-Enrich performs gene set enrichment testing
using peaks called from a ChIP-seq experiment. The method empirically
corrects for confounding factors such as the length of genes,
and the mappability of the sequence surrounding genes.

biocViews Software, Bioinformatics, Enrichment, GeneSetEnrichment

License GPL-3

Imports chipenrich.data, GenomeInfoDb, GenomicRanges, grid, IRanges,
lattice, latticeExtra, methods, mgcv, parallel, plyr, rms,
S4Vectors, stringr

Suggests BiocStyle, devtools, knitr, rmarkdown, roxygen2, testthat

Depends R (>= 3.1.0)

LazyLoad yes

Maintainer Raymond G. Cavalcante <rcavalca@umich.edu>

RoxygenNote 5.0.1

VignetteBuilder knitr

Collate 'assign_peaks.R' 'test_gam.R' 'test_fisher.R'
'test_binomial.R' 'test_approx.R' 'peaks_per_gene.R' 'read.R'
'randomize.R' 'setup.R' 'supported.R' 'utils.R' 'constants.R'
'plot_spline_length.R' 'plot_gene_coverage.R'
'plot_dist_to_tss.R' 'chipenrich.R' 'chipenrich_package_doc.R'

NeedsCompilation no

Author Ryan P. Welch [aut, cph],
Chee Lee [aut],
Raymond G. Cavalcante [aut, cre],
Laura J. Scott [ths],
Maureen A. Sartor [ths]

R topics documented:

`assign_peaks` *Assign peak midpoints to defined gene loci.*

Description

Determine the midpoints of a set of input regions `peaks` and the overlap of the midpoints with a given locus definition `locusdef`. Also report the TSS that is nearest each region (peak) overlapping a defined locus and its distance.

Usage

```
assign_peaks(peaks, locusdef, tss)
```

Arguments

`peaks`	A `GRanges` object representing regions to be used for enrichment.
`locusdef`	A locus definition object from `chipenrich.data`.
`tss`	A tss object from `chipenrich.data`

Details

Typically, this function will not be used alone, but inside `chipenrich()`.

Value

A `data.frame` with columns for `peak_id, chr, peak_start, peak_end, gene_locus_start, gene_locus_end, g`
The result is used in `num_peaks_per_gene()`.

Examples

```
data('locusdef.hg19.nearest_tss', package = 'chipenrich.data')
data('tss.hg19', package = 'chipenrich.data')

file = system.file('extdata', 'test_assign.bed', package = 'chipenrich')
peaks = read_bed(file)

assigned_peaks = assign_peaks(
peaks = peaks,
locusdef = locusdef.hg19.nearest_tss,
tss = tss.hg19)
```

`assign_peak_segments` *Assign whole peaks to all overlapping defined gene loci.*

Description

Determine all overlaps between the set of input regions `peaks` and the given locus definition `locusdef`. In addition, report where each overlap begins and ends, as well as the length of the overlap.

Usage

```
assign_peak_segments(peaks, locusdef)
```

Arguments

`peaks`	A `GRanges` object representing regions to be used for enrichment.
`locusdef`	A locus definition object from `chipenrich.data`.

Details

Typically, this function will not be used alone, but inside `chipenrich()` with `method = 'broadenrich'`.

Value

A `data.frame` with columns for `peak_id, chr, peak_start, peak_end, gene_locus_start, gene_locus_end, g`
The result is used in `num_peaks_per_gene()`.

Examples

```
data('locusdef.hg19.nearest_tss', package = 'chipenrich.data')
data('tss.hg19', package = 'chipenrich.data')

file = system.file('extdata', 'test_assign.bed', package = 'chipenrich')
peaks = read_bed(file)

assigned_peaks = assign_peak_segments(
peaks = peaks,
locusdef = locusdef.hg19.nearest_tss)
```

calc_peak_gene_overlap *Add peak overlap and ratio to result of* num_peaks_per_gene()

Description

In particular, for method = 'broadenrich' in chipenrich(), when using assign_peak_segments(). This function will add aggregated peak_overlap (in base pairs) and ratio (relative to length) columns to the result of num_peaks_per_gene() so the right data is present for the method = 'broadenrich' model.

Usage

```
calc_peak_gene_overlap(assigned_peaks, ppg)
```

Arguments

assigned_peaks	A data.frame resulting from assign_peak_segments().
ppg	The aggregated peak assignments over geneid from num_peaks_per_gene().

Details

Typically, this function will not be used alone, but inside chipenrich() with method = 'broadenrich'.

Value

A data.frame with columns geneid, length, log10_length, num_peaks, peak, peak_overlap, ratio. The result is used directly in the gene set enrichment tests in chipenrich() when method = 'broadenrich'.

Examples

```
data('locusdef.hg19.nearest_tss', package = 'chipenrich.data')
data('tss.hg19', package = 'chipenrich.data')

file = system.file('extdata', 'test_assign.bed', package = 'chipenrich')
peaks = read_bed(file)

assigned_peaks = assign_peak_segments(
peaks = peaks,
locusdef = locusdef.hg19.nearest_tss)

ppg = num_peaks_per_gene(
assigned_peaks = assigned_peaks,
locusdef = locusdef.hg19.nearest_tss,
mappa = NULL)

ppg = calc_peak_gene_overlap(
assigned_peaks = assigned_peaks,
ppg = ppg)
```

chipenrich *Run ChIP-Enrich on a dataset of ChIP-seq peaks*

Description

Run gene set enrichment testing (ChIP-Enrich or Broad-Enrich) on a ChIP-seq peak dataset or other type of dataset consisting of regions across the genome. The user can call `chipenrich` to run the method on their data. A number of arguments can be provided to change the type of test, the genome build, which sets of genes to test, how peaks are assigned to genes, and other minor options.

ChIP-Enrich performs gene set enrichment testing using peaks called from a ChIP-seq experiment. The method empirically corrects for confounding factors such as the length of genes, and the mappability of the sequence surrounding genes.

Usage

```
chipenrich(peaks, out_name = "chipenrich", out_path = getwd(),
  genome = "hg19", genesets = c("GOBP", "GOCC", "GOMF"),
  locusdef = "nearest_tss", method = "chipenrich",
  fisher_alt = "two.sided", use_mappability = F, mappa_file = NULL,
  read_length = 36, qc_plots = T, max_geneset_size = 2000,
  num_peak_threshold = 1, n_cores = 1)
```

Arguments

`peaks`	A `data.frame`, or tab-delimited text file (BED, narrowPeak, broadPeak, etc) with the first three columns being chrom, start, and end. The data frame should have at least 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX".
`out_name`	Prefix string to use for naming output files. This should not contain any characters that would be illegal for the system being used (Unix, Windows, etc.) The default value is "chipenrich", and a file "chipenrich_results.tab" is produced. If `qc_plots` is set, then a file "chipenrich_qcplots.pdf" is produced containing a number of quality control plots. If `out_name` is set to NULL, no files are written, and results then must be retrieved from the list returned by `chipenrich`.
`out_path`	Directory to which results files will be written out. Defaults to the current working directory as returned by `getwd`.
`genome`	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the `supported_genomes` function.
`genesets`	A character vector of geneset databases to be tested for enrichment. A list of supported geneset databases can be generated by the '`supported_genesets` function.
`locusdef`	A string denoting the gene locus definition to be used, or the full path to a user-defined locus definition file. A gene locus definition controls how peaks are assigned to genes. See `supported_locusdefs` for a list of supported definitions built-in. If using a user-specified file, the file must have 4 columns: geneid, chrom, start, end and be tab-delimited.
`method`	A character string specifying the method to use for enrichment testing. Must be one of ChIP-Enrich ('chipenrich') (default), Broad-Enrich ('broadenrich'), or Fisher's exact test ('fet'). For a list of supported methods, use `supported_methods`.

`fisher_alt` If method is 'fet', this option indicates the alternative for Fisher's exact test, and must be one of 'two-sided' (default), 'greater', or 'less'.

`use_mappability` A logical variable indicating whether to adjust for mappability. If enabled, this option will use our internally calculated mappabilities for each gene locus given the length of reads used in the experiment (see `read_length` option). NOTE: If providing a `mappa_file` this parameter should be set to FALSE.

`mappa_file` Path to a file containing user-specified gene locus mappability. The file should contain two columns: geneid and mappa. Gene IDs should be Entrez gene IDs. Mappability values should be between 0 and 1.

`read_length` If adjusting for mappability, this number specifies the read length to be used. The read length given here should ideally correspond to the length of reads from the original experiment.

`qc_plots` A logical variable that enables the automatic generation of plots for quality control.

`max_geneset_size` Sets the maximum number of genes a gene set may have to be considered for enrichment testing.

`num_peak_threshold` Sets the threshold for how many peaks a gene must have to be considered as having a peak. Defaults to 1. Only relevant for Fisher's exact test and ChIP-Enrich methods.

`n_cores` The number of cores to use for enrichment testing. We recommend using only up to the maximum number of *physical* cores present, as virtual cores do not significantly decrease runtime. Default number of cores is set to 1. NOTE: Windows does not support multicore enrichment.

Value

A list, containing the following items:

`peaks` A data frame containing peak assignments to genes. Peaks which do not overlap a gene locus are not included. Each peak that was assigned to a gene is listed, along with the peak midpoint or peak interval coordinates (depending on which was used), the gene to which the peak was assigned, the locus start and end position of the gene, and the distance from the peak to the TSS.

The columns are:

peak_id is an ID given to unique combinations of chromosome, peak start, and peak end.

chrom is the chromosome the peak originated from.

peak_start is start position of the peak.

peak_end is end position of the peak.

peak_midpoint is the midpoint of the peak.

geneid is the Entrez ID of the gene to which the peak was assigned.

gene_symbol is the official gene symbol for the geneid (above).

gene_locus_start is the start position of the locus for the gene to which the peak was assigned (specified by the locus definition used.)

gene_locus_end is the end position of the locus for the gene to which the peak was assigned (specified by the locus definition used.)

nearest_tss (`method='chipenrich'` and `method='fet'`) is the closest TSS to this peak (for any gene, not necessarily the gene this peak was assigned to.)

nearest_tss_gene (`method='chipenrich'` and `method='fet'`) is the gene having the closest TSS to the peak (should be the same as geneid when using the nearest TSS locus definition.)

nearest_tss_gene_strand (`method='chipenrich'` and `method='fet'`) is the strand of the gene with the closest TSS.

overlap_start (`method='broadenrich'` only) the start position of the peak overlap with the gene locus.

overlap_end (`method='broadenrich'` only) the end position of the peak overlap with the gene locus.

peak_overlap (`method='broadenrich'` only) the base pair overlap of the peak with the gene locus.

`results` A data frame of the results from performing the gene set enrichment test on each geneset that was requested (all genesets are merged into one final data frame.) The columns are:

Geneset.ID is the identifier for a given gene set from the selected database. For example, GO:0000003.

Geneset.Type specifies from which database the Geneset.ID originates. For example, "Gene Ontology Biological Process."

Description gives a definition of the geneset. For example, "reproduction."

P.Value is the probability of observing the degree of enrichment of the gene set given the null hypothesis that peaks are not associated with any gene sets.

FDR is the false discovery rate proposed by Bejamini \& Hochberg for adjusting the p-value to control for family-wise error rate.

Odds.Ratio is the estimated odds that peaks are associated with a given gene set compared to the odds that peaks are associated with other gene sets, after controlling for locus length and/or mappability. An odds ratio greater than 1 indicates enrichment, and less than 1 indicates depletion.

N.Geneset.Genes is the number of genes in the gene set.

N.Geneset.Peak.Genes is the number of genes in the genes set that were assigned at least one peak.

Geneset.Avg.Gene.Length is the average length of the genes in the gene set.

Geneset.Avg.Gene.Coverage (`method='broadenrich'` only) is the mean proportion of the gene loci in the gene set covered by a peak.

Geneset.Peak.Genes is the list of genes from the gene set that had at least one peak assigned.

`opts` A data frame containing the arguments/values passed to `chipenrich`.

`peaks_per_gene`

A data frame of the count of peaks per gene. The columns are:

geneid is the Entrez Gene ID.

length is the length of the gene's locus (depending on which locus definition you chose.)

log10_length is the log10(locus length) for the gene.

num_peaks is the number of peaks that were assigned to the gene, given the current locus definition.

peak is whether or not the gene is considered to have a peak, as defined by `num_peak_threshold`.

peak_overlap (method='broadenrich' only) is the number of base pairs of the gene covered by a peak.

ratio (method='broadenrich' only) is the proportion of the gene covered by a peak.

Examples

```
# Run ChipEnrich using an example dataset, assigning peaks to the nearest TSS,
# testing all Biocarta and Panther pathways
data(peaks_E2F4, package = 'chipenrich.data')
peaks_E2F4 = subset(peaks_E2F4, peaks_E2F4$chrom == 'chr1')
gs_path = system.file('extdata','vignette_genesets.txt', package='chipenrich')
results = chipenrich(peaks_E2F4, method='chipenrich', locusdef='nearest_tss',
genesets=gs_path, out_name=NULL)

# Get the list of peaks that were assigned to genes.
assigned_peaks = results$peaks

# Get the results of enrichment testing.
enrich = results$results
```

---

load_peaks *Convert BEDX+Y data.frames and into GRanges*

Description

Given a data.frame in BEDX+Y format, keep only the first three columns: chrom, start, end, and output GRanges.

Usage

```
load_peaks(dframe)
```

Arguments

dframe	A BEDX+Y style data.frame.

Details

Typically, this function will not be used alone, but inside chipenrich().

Value

A GRanges that is unstranded, and contains only chrom, start, and end.

Examples

```
# Example of BED3 with no header
data(peaks_H3K4me3_GM12878, package='chipenrich.data')
peaks_H3K4me3_GM12878 = subset(peaks_H3K4me3_GM12878, peaks_H3K4me3_GM12878$chrom == 'chr1')
peaks = load_peaks(peaks_H3K4me3_GM12878)
```

num_peaks_per_gene *Aggregate peak assignments over the* geneid *column*

Description

For each geneid, determine the locus length and the number of peaks.

Usage

```
num_peaks_per_gene(assigned_peaks, locusdef, mappa = NULL)
```

Arguments

assigned_peaks	A data.frame resulting from assign_peaks() or assign_peak_segments().
locusdef	A locus definition object from chipenrich.data.
mappa	A mappability object from chipenrich.data.

Details

Typically, this function will not be used alone, but inside chipenrich().

Value

A data.frame with columns geneid, length, log10_length, num_peaks, peak. The result is used directly in the gene set enrichment tests in chipenrich().

Examples

```
data('locusdef.hg19.nearest_tss', package = 'chipenrich.data')
data('tss.hg19', package = 'chipenrich.data')

file = system.file('extdata', 'test_assign.bed', package = 'chipenrich')
peaks = read_bed(file)

assigned_peaks = assign_peaks(
peaks = peaks,
locusdef = locusdef.hg19.nearest_tss,
tss = tss.hg19)

ppg = num_peaks_per_gene(
assigned_peaks = assigned_peaks,
locusdef = locusdef.hg19.nearest_tss,
mappa = NULL)
```

plot_dist_to_tss *Plot histogram of distance from peak to nearest TSS*

Description

Create a histogram of the distance from each peak to the nearest transcription start site (TSS) of any gene.

Usage

```
plot_dist_to_tss(peaks, genome = "hg19")
```

Arguments

peaks	A data.frame, or tab-delimited text file (BED, narrowPeak, broadPeak, etc) with the first three columns being chrom, start, and end. The data frame should have at least 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX".
genome	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the supported_genomes function.

Value

A trellis plot object.

Examples

```
# Create histogram of distance from peaks to nearest TSS.
data(peaks_E2F4, package = 'chipenrich.data')
peaks_E2F4 = subset(peaks_E2F4, peaks_E2F4$chrom == 'chr1')
plot_dist_to_tss(peaks_E2F4)
```

plot_gene_coverage *Plot probability of peak being assigned to a gene vs. gene length*

Description

Create a plot showing the probability of a gene being assigned a peak given its locus length. The plot shows an empirical fit to the data using a binomial smoothing spline.

Usage

```
plot_gene_coverage(peaks, locusdef = "nearest_tss", genome = "hg19",
  use_mappability = F, read_length = 36, legend = T, xlim = NULL)
```

Arguments

peaks	A data.frame, or tab-delimited text file (BED, narrowPeak, broadPeak, etc) with the first three columns being chrom, start, and end. The data frame should have at least 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX".
locusdef	A string denoting the gene locus definition to be used, or the full path to a user-defined locus definition file. A gene locus definition controls how peaks are assigned to genes. See supported_locusdefs for a list of supported definitions built-in. If using a user-specified file, the file must have 4 columns: geneid, chrom, start, end and be tab-delimited.
genome	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the supported_genomes function.
use_mappability	A logical variable indicating whether to adjust for mappability. If enabled, this option will use our internally calculated mappabilities for each gene locus given the length of reads used in the experiment (see read_length option). NOTE: If providing a mappa_file this parameter should be set to FALSE.
read_length	If adjusting for mappability, this number specifies the read length to be used. The read length given here should ideally correspond to the length of reads from the original experiment.
legend	If true, a legend will be drawn on the plot.
xlim	Set the x-axis limit. NULL means select x-lim automatically.

Value

A trellis plot object.

Examples

```
# Spline plot for E2F4 example peak dataset.
data(peaks_E2F4, package = 'chipenrich.data')
peaks_E2F4 = subset(peaks_E2F4, peaks_E2F4$chrom == 'chr1')
plot_gene_coverage(peaks_E2F4)
```

---

plot_spline_length *Plot probability of peak being assigned to a gene vs. gene length*

Description

Create a plot showing the probability of a gene being assigned a peak given its locus length. The plot shows an empirical fit to the data using a binomial smoothing spline.

Usage

```
plot_spline_length(peaks, locusdef = "nearest_tss", genome = "hg19",
  use_mappability = F, read_length = 36, legend = T, xlim = NULL)
```

Arguments

peaks	A data.frame, or tab-delimited text file (BED, narrowPeak, broadPeak, etc) with the first three columns being chrom, start, and end. The data frame should have at least 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX".
locusdef	A string denoting the gene locus definition to be used, or the full path to a user-defined locus definition file. A gene locus definition controls how peaks are assigned to genes. See supported_locusdefs for a list of supported definitions built-in. If using a user-specified file, the file must have 4 columns: geneid, chrom, start, end and be tab-delimited.
genome	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the supported_genomes function.
use_mappability	A logical variable indicating whether to adjust for mappability. If enabled, this option will use our internally calculated mappabilities for each gene locus given the length of reads used in the experiment (see read_length option). NOTE: If providing a mappa_file this parameter should be set to FALSE.
read_length	If adjusting for mappability, this number specifies the read length to be used. The read length given here should ideally correspond to the length of reads from the original experiment.
legend	If true, a legend will be drawn on the plot.
xlim	Set the x-axis limit. NULL means select x-lim automatically.

Value

A trellis plot object.

Examples

```
# Spline plot for E2F4 example peak dataset.
data(peaks_E2F4, package = 'chipenrich.data')
peaks_E2F4 = subset(peaks_E2F4, peaks_E2F4$chrom == 'chr1')

plot_spline_length(peaks_E2F4,genome='hg19')

# Create the plot for a different locus definition
# to compare the effect.
plot_spline_length(peaks_E2F4, locusdef = 'nearest_gene', genome = 'hg19')
```

---

read_bed *Read BEDX+Y files and convert into GRanges*

Description

Given a file_path, read in a delimited file, assuming it is BEDX+Y, keep only the first three columns: chrom, start, end, and output GRanges.

Usage

```
read_bed(file_path)
```

Arguments

file_path	A path to a valid BEDX+Y file.

Details

Typically, this function will not be used alone, but inside chipenrich().

Value

A GRanges that is unstranded, and contains only chrom, start, and end.

Examples

```
# Example of BED3 with no header
file = system.file('extdata', 'test_assign.bed', package = 'chipenrich')
peaks = read_bed(file)

# Example of BED3 with header
file = system.file('extdata', 'test_header.bed', package = 'chipenrich')
peaks = read_bed(file)

# Example of narrowPeak with header
file = system.file('extdata', 'test.narrowPeak', package = 'chipenrich')
peaks = read_bed(file)
```

---

read_bedgff *Read BEDGFF files and convert into GRanges*

Description

Given a file_path, read in a delimited file, assuming it is BEDGFF (as is output by modENCODE for D. Melanogaster TF ChIP-seq experiments), keep only chrom, start, and end columns, and output GRanges.

Usage

```
read_bedgff(file_path)
```

Arguments

file_path	A path to a valid BEDGFF file (as from modENCODE).

Details

Typically, this function will not be used alone, but inside chipenrich().

Value

A GRanges that is unstranded, and contains only chrom, start, and end.

Examples

```
# Example of GFF3
file = system.file('extdata', 'test.gff3', package = 'chipenrich')
peaks = read_bedgff(file)

# Example of gzipped GFF3
file = system.file('extdata', 'test.gff3.gz', package = 'chipenrich')
peaks = read_bedgff(file)
```

---

supported_genesets *Display supported genesets for gene set enrichment.*

Description

Display supported genesets for gene set enrichment.

Usage

```
supported_genesets()
```

Value

A data.frame with columns geneset, organism.

Examples

```
supported_genesets()
```

---

supported_genomes *Display supported genomes.*

Description

Display supported genomes.

Usage

```
supported_genomes()
```

Value

A vector indicating supported genomes.

Examples

```
supported_genomes()
```

`supported_locusdefs` *Display supported locus definitions*

Description

Display supported locus definitions

Usage

```
supported_locusdefs()
```

Value

A `data.frame` with columns `genome, locusdef`.

Examples

```
supported_locusdefs()
```

`supported_methods` *Display supported gene set enrichment methods.*

Description

Display supported gene set enrichment methods.

Usage

```
supported_methods()
```

Value

A vector indicating supported methods for gene set enrichment.

Examples

```
supported_methods()
```

`supported_read_lengths` *Display supported read lengths for mappability*

Description

Display supported read lengths for mappability

Usage

```
supported_read_lengths()
```

Value

A `data.frame` with columns `genome, locusdef, read_length`.

Examples

```
supported_read_lengths()
```

Index