Package 'chipenrich'

April 11, 2018

Type Package

Title Gene Set Enrichment For ChIP-seq Peak Data

Version 2.2.0

Date 2017-10-27

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 ChIPSeq, Epigenetics, FunctionalGenomics, GeneSetEnrichment, HistoneModification, Regression

License GPL-3

Imports AnnotationDbi, BiocGenerics, chipenrich.data, GenomeInfoDb, GenomicRanges, grDevices, grid, IRanges, lattice, latticeExtra, methods, mgcv, org.Dm.eg.db, org.Dr.eg.db, org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db, parallel, plyr, rms, rtracklayer, S4Vectors, stats, stringr, utils

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

Depends R (>= 3.4.0)

LazyLoad yes

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

RoxygenNote 6.0.1

VignetteBuilder knitr

Collate 'assign_peaks.R' 'test_broadenrich.R' 'peaks_per_gene.R' 'read.R' 'randomize.R' 'setup.R' 'supported.R' 'utils.R' 'constants.R' 'plot_gene_coverage.R' 'broadenrich.R' 'test_chipapprox.R' 'test_chipenrich.R' 'test_chipenrich_slow.R' 'test_fisher.R' 'test_binomial.R' 'test_approx.R' 'plot_chipenrich_spline.R' 'plot_dist_to_tss.R' 'chipenrich.R' 'chipenrich_package_doc.R' 'test_polyapprox.R' 'test_polyenrich_weighted.R' 'test_polyenrich_slow.R' 'test_polyenrich.R' 'plot_polyenrich_spline.R' 'polyenrich.R' 'hybrid.R' 'peak_weights.R'

NeedsCompilation no

Author Ryan P. Welch [aut, cph],
Chee Lee [aut],
Raymond G. Cavalcante [aut, cre],
Chris Lee [aut],
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, weighting = NULL)
```

Arguments

peaks	A GRanges object representing regions to be used for enrichment.
locusdef	A locus definition object from chipenrich.data.
tss	A GRanges object representing the TSSs for the genome build. Includes mcols for Entrez Gene ID gene_id and gene symbol symbol.
weighting	A string defining what weighting option they want. Current options are 'multiAssign', 'signalValue', and 'logSignal Value'. Default is NULL.

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)
```

broadenrich *Run Broad-Enrich on broad genomic regions*

Description

Broad-Enrich is designed for use with broad peaks that may intersect multiple gene loci, and cumulatively cover greater than 5% of the genome. For example, ChIP-seq experiments for histone modifications. For more details, see the 'Broad-Enrich Method' section below. For help choosing a method, see the 'Choosing A Method' section below, or see the vignette.

Usage

```
broadenrich(peaks, out_name = "broadenrich", out_path = getwd(),
  genome = supported_genomes(), genesets = c("GOBP", "GOCC", "GOMF"),
  locusdef = "nearest_tss", mappability = NULL, qc_plots = TRUE,
  min_geneset_size = 15, max_geneset_size = 2000, randomization = NULL,
  n_cores = 1)
```

Arguments

peaks	Either a file path or a data.frame of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a data.frame A BEDX+Y style data.frame. See GenomicRanges::makeGRangesFromDataFrame for acceptable column names.
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 "broadenrich", and a file "broadenrich_results.tab" is produced. If qc_plots is set, then a file "broadenrich_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 broadenrich.
out_path	Directory to which results files will be written out. Defaults to the current working directory as returned by getwd.
genome	One of the supported_genomes().
genesets	A character vector of geneset databases to be tested for enrichment. See supported_genesets(). Alternately, a file path to a a tab-delimited text file with header and first column being the geneset ID or name, and the second column being Entrez Gene IDs. For an example custom gene set file, see the vignette.
locusdef	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or supported_locusdefs. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
mappability	One of NULL, a file path to a custom mappability file, or an integer for a valid read length given by supported_read_lengths. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.
qc_plots	A logical variable that enables the automatic generation of plots for quality control.
min_geneset_size	Sets the minimum number of genes a gene set may have to be considered for enrichment testing.
max_geneset_size	Sets the maximum number of genes a gene set may have to be considered for enrichment testing.

`randomization` One of `NULL`, 'complete', 'bylength', or 'bylocation'. See the Randomizations section below.

`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:

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

`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.

chr is the chromosome the peak originated from.

peak_start is start position of the peak.

peak_end is end position of the peak.

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

gene_symbol is the official gene symbol for the gene_id (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.)

overlap_start the start position of the peak overlap with the gene locus.

overlap_end the end position of the peak overlap with the gene locus.

peak_overlap the base pair overlap of the peak with the gene locus.

`peaks_per_gene`

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

gene_id 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 is the number of base pairs of the gene covered by a peak.

ratio is the proportion of the gene covered by a peak.

`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 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.

Broad-Enrich Method

The Broad-Enrich method uses the cumulative peak coverage of genes in its model for enrichment: `GO ~ ratio + s(log10_length)`. Here, `GO` is a binary vector indicating whether a gene is in the gene set being tested, `ratio` is a numeric vector indicating the ratio of the gene covered by peaks, and `s(log10_length)` is a binomial cubic smoothing spline which adjusts for the relationship between gene coverage and locus length.

Choosing A Method

The following guidelines are intended to help select an enrichment function:

broadenrich(): is designed for use with broad peaks that may intersect multiple gene loci, and cumulatively cover greater than 5% of the genome. For example, ChIP-seq experiments for histone modifications.

chipenrich(): is designed for use with 1,000s or 10,000s of narrow peaks which results in fewer gene loci containing a peak overall. For example, ChIP-seq experiments for transcription factors.

polyenrich(): is also designed for narrow peaks, but where there are 100,000s of peaks which results in nearly every gene locus containing a peak. For example, ChIP-seq experiments for transcription factors.

Randomizations

Randomization of locus definitions allows for the assessment of Type I Error under the null hypothesis. The randomization codes are:

`NULL`**:** No randomizations, the default.

'complete': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together, without regard for the chromosome location, or locus length. The null hypothesis is that there is no true gene set enrichment.

'bylength': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together within bins of 100 genes sorted by locus length. The null hypothesis is that there is no true gene set enrichment, but with preserved locus length relationship.

'bylocation': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together within bins of 50 genes sorted by genomic location. The null hypothesis is that there is no true gene set enrichment, but with preserved genomic location.

The return value with a selected randomization is the same list as without. To assess the Type I error, the `alpha` level for the particular data set can be calculated by dividing the total number of gene sets with p-value < `alpha` by the total number of tests. Users may want to perform multiple randomizations for a set of peaks and take the median of the `alpha` values.

See Also

Other enrichment functions: `chipenrich`, `polyenrich`

Examples

```
# Run Broad-Enrich using an example dataset, assigning peaks to the nearest TSS,
# and on a small custom geneset
data(peaks_H3K4me3_GM12878, package = 'chipenrich.data')
peaks_H3K4me3_GM12878 = subset(peaks_H3K4me3_GM12878,
peaks_H3K4me3_GM12878$chrom == 'chr1')
gs_path = system.file('extdata','vignette_genesets.txt', package='chipenrich')
results = broadenrich(peaks_H3K4me3_GM12878, locusdef='nearest_tss',
genome = 'hg19', 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
```

---

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 gene_id 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 gene_id, 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 narrow genomic regions*

Description

ChIP-Enrich is designed for use with 1,000s or 10,000s of narrow peaks which results in fewer gene loci containing a peak overall. For example, ChIP-seq experiments for transcription factors. For more details, see the 'ChIP-Enrich Method' section below. For help choosing a method, see the 'Choosing A Method' section below, or see the vignette.

Usage

```
chipenrich(peaks, out_name = "chipenrich", out_path = getwd(),
  genome = supported_genomes(), genesets = c("GOBP", "GOCC", "GOMF"),
  locusdef = "nearest_tss", method = "chipenrich", mappability = NULL,
  fisher_alt = "two.sided", qc_plots = TRUE, min_geneset_size = 15,
  max_geneset_size = 2000, num_peak_threshold = 1, randomization = NULL,
  n_cores = 1)
```

Arguments

peaks	Either a file path or a data.frame of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a data.frame A BEDX+Y style data.frame. See GenomicRanges::makeGRangesFromDataFrame for acceptable column names.
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	One of the supported_genomes().
genesets	A character vector of geneset databases to be tested for enrichment. See supported_genesets(). Alternately, a file path to a a tab-delimited text file with header and first column being the geneset ID or name, and the second column being Entrez Gene IDs. For an example custom gene set file, see the vignette.
locusdef	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or supported_locusdefs. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
method	A character string specifying the method to use for enrichment testing. Must be one of ChIP-Enrich ('chipenrich') (default), or Fisher's exact test ('fet'). For a list of supported methods, use supported_methods().
mappability	One of NULL, a file path to a custom mappability file, or an integer for a valid read length given by supported_read_lengths. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.
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'.
qc_plots	A logical variable that enables the automatic generation of plots for quality control.
min_geneset_size	Sets the minimum number of genes a gene set may have to be considered for enrichment testing.
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.

`randomization`
: One of `NULL`, 'complete', 'bylength', or 'bylocation'. See the Randomizations section below.

`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:

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

`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.

  chr 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.

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

  gene_symbol is the official gene symbol for the gene_id (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 is the closest TSS to this peak (for any gene, not necessarily the gene this peak was assigned to.)

  nearest_tss_gene is the gene having the closest TSS to the peak (should be the same as gene_id when using the nearest TSS locus definition.)

  nearest_tss_gene_strand is the strand of the gene with the closest TSS.

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

  gene_id 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`.

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

ChIP-Enrich Method

The ChIP-Enrich method uses the presence of a peak in its model for enrichment: `peak ~ GO + s(log10_length)`. Here, `GO` is a binary vector indicating whether a gene is in the gene set being tested, `peak` is a binary vector indicating the presence of a peak in a gene, and `s(log10_length)` is a binomial cubic smoothing spline which adjusts for the relationship between the presence of a peak and locus length.

Choosing A Method

The following guidelines are intended to help select an enrichment function:

broadenrich(): is designed for use with broad peaks that may intersect multiple gene loci, and cumulatively cover greater than 5% of the genome. For example, ChIP-seq experiments for histone modifications.

chipenrich(): is designed for use with 1,000s or 10,000s of narrow peaks which results in fewer gene loci containing a peak overall. For example, ChIP-seq experiments for transcription factors.

polyenrich(): is also designed for narrow peaks, but where there are 100,000s of peaks which results in nearly every gene locus containing a peak. For example, ChIP-seq experiments for transcription factors.

Randomizations

Randomization of locus definitions allows for the assessment of Type I Error under the null hypothesis. The randomization codes are:

`NULL`**:** No randomizations, the default.

'complete': Shuffle the gene_id and symbol columns of the locusdef together, without regard for the chromosome location, or locus length. The null hypothesis is that there is no true gene set enrichment.

'bylength': Shuffle the gene_id and symbol columns of the locusdef together within bins of 100 genes sorted by locus length. The null hypothesis is that there is no true gene set enrichment, but with preserved locus length relationship.

'bylocation': Shuffle the gene_id and symbol columns of the locusdef together within bins of 50 genes sorted by genomic location. The null hypothesis is that there is no true gene set enrichment, but with preserved genomic location.

The return value with a selected randomization is the same list as without. To assess the Type I error, the alpha level for the particular data set can be calculated by dividing the total number of gene sets with p-value < alpha by the total number of tests. Users may want to perform multiple randomizations for a set of peaks and take the median of the alpha values.

See Also

Other enrichment functions: broadenrich, polyenrich

Examples

```
# Run ChipEnrich using an example dataset, assigning peaks to the nearest TSS,
# and on a small custom geneset
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',
genome = 'hg19', 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
```

---

chipenrich_package *chipenrich: Gene Set Enrichment For ChIP-seq Peak Data and Other Genomic Regions*

Description

The chipenrich package includes three classes of methods that adjust for potential confounders of gene set enrichment testing (locus length and mappability of the sequence reads). The first, chipenrich, is designed for use with transcription-factor (TF) based ChIP-seq experiments and other DNA sequencing experiments with narrow genomic regions. The second, polyenrich, is similarly designed for TF based ChIP-seq, but where the number of peaks present in gene loci may be important. The third, broadenrich, is designed for use with histone modification based ChIP-seq experiments and other DNA sequencing experiments with broad genomic regions.

filter_genesets *Function to filter genesets by locus definition and size*

Description

This function filters gene sets based on the genes that are present in a particular locus definition. After determining which genes are present in both the GeneSet, `gs_obj`, and the LocusDefinition `ldef_obj`, gene sets are filtered by size with `min_geneset_size` and `max_geneset_size`.

Usage

```
filter_genesets(gs_obj, ldef_obj, min_geneset_size = 15,
  max_geneset_size = 2000)
```

Arguments

`gs_obj`	A valid GeneSet object
`ldef_obj`	A valid LocusDefinition object
`min_geneset_size`	An integer indicating the floor for genes in a geneset. Default 15.
`max_geneset_size`	An integer indicating the ceiling for genes in a geneset. Default 2000.

Value

An altered `gs_obj` with changed `set.gene` and `all.genes` slots reflecting `min_geneset_size` and `max_geneset_size` after intersecting with the genes present in the particular locus definition.

genome_to_organism *Get the correct organism code based on genome*

Description

Data from `chipenrich.data` uses three letter organism codes for the `GeneSet` objects. This function ensures the correct objects are loaded.

Usage

```
genome_to_organism(genome = supported_genomes())
```

Arguments

`genome`	One of the `supported_genomes()`.

Value

A string for the three letter organism code. Convention is first letter of the first word in the binomial name, and first two letters of the second word in the binomial name. 'Homo sapiens' is then 'hsa', for example.

genome_to_orgdb *Get Entrez ID to gene symbol mappings for custom locus definitions*

Description

If a custom locus definition is one of the `supported_genomes()`, then the gene symbol column of the custom locus definition is populated using the appropriate orgDb package.

Usage

```
genome_to_orgdb(genome = supported_genomes())
```

Arguments

`genome`	One of the `supported_genomes()`.

Value

A `data.frame` with `gene_id` and `symbol` columns.

get_test_method *Get the test function name from the method name*

Description

The `method` comes from what is used in `chipenrich()` or in `polyenrich()`.

Usage

```
get_test_method(method)
```

Arguments

`method`	A character for the method used. One of the `supported_methods` or one of the `HIDDEN_METHODS` in `constants.R`.

Value

A singleton named character vector with value of the test function and name of the method.

hybridenrich *Running Hybrid test, either from scratch or using two results files*

Description

Hybrid test is designed for people unsure of which test between ChIP-Enrich and Poly-Enrich to use, so it takes information of both and gives adjusted P-values. For more about ChIP- and Poly-Enrich, consult their corresponding documentation.

Usage

```
hybridenrich(peaks, out_name = "hybridenrich", out_path = getwd(),
  genome = supported_genomes(), genesets = c("GOBP", "GOCC", "GOMF"),
  locusdef = "nearest_tss", methods = c("chipenrich", "polyenrich"),
  weighting = NULL, mappability = NULL, qc_plots = TRUE,
  min_geneset_size = 15, max_geneset_size = 2000, num_peak_threshold = 1,
  randomization = NULL, n_cores = 1)
```

Arguments

peaks	Either a file path or a data.frame of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a data.frame A BEDX+Y style data.frame. See GenomicRanges::makeGRangesFromDataFrame for acceptable column names.
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	One of the supported_genomes().
genesets	A character vector of geneset databases to be tested for enrichment. See supported_genesets(). Alternately, a file path to a a tab-delimited text file with header and first column being the geneset ID or name, and the second column being Entrez Gene IDs. For an example custom gene set file, see the vignette.
locusdef	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or supported_locusdefs. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.

`methods` A character string array specifying the method to use for enrichment testing. Currently actually unused as the methods are forced to be one chipenrich and one polyenrich.

`weighting` A character string specifying the weighting method. Method name will automatically be "polyenrich_weighted" if given weight options. Current options are: 'signalValue', 'logsignalValue', and 'multiAssign'.

`mappability` One of `NULL`, a file path to a custom mappability file, or an `integer` for a valid read length given by `supported_read_lengths`. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.

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

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

`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.

`randomization` One of `NULL`, 'complete', 'bylength', or 'bylocation'. See the Randomizations section below.

`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 data.frame containing:

`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.

P.Value.x 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, for the first test..

P.Value.y is the same as above except for the second test.

P.Value.Hybrid The calculated Hybrid p-value from the two tests

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

Hybrid p-values

Given n tests that test for the same hypothesis, same Type I error rate, and converted to p-values: `p_1, ..., p_n`, the Hybrid p-value is computed as: `n*min(p_1, ..., p_n)`. This hybrid test will

have at most the same Type I error as any individual test, and if any of the tests have 100 sample size goes to infinity, then so will the hybrid test.

Function inputs

Every input in hybridenrich is the same as in chipenrich and polyenrich. Inputs unique to chipenrich are: num_peak_threshold; and inputs unique to polyenrich are: weighting. Currently the test only supports running chipenrich and polyenrich, but future plans will allow you to run any number of different support tests.

Joining two results files

Combines two existing results files and returns one results file with hybrid p-values and FDR included. Current allowed inputs are objects from any of the supplied enrichment tests or a dataframe with at least the following columns: `P.value, Geneset.ID`. Optional columns include: `Status`. Currently we only allow for joining two results files, but future plans will allow you to join any number of results files.

---

load_peaks *Convert a BEDX+Y data.frame and into GRanges*

Description

Given a `data.frame` in BEDX+Y format, use the built-in function `GenomicRanges::makeGRangesFromDataFrame()` to convert to `GRanges`.

Usage

```
load_peaks(dframe, keep.extra.columns = TRUE)
```

Arguments

`dframe`	A BEDX+Y style `data.frame`. See `GenomicRanges::makeGRangesFromDataFrame` for acceptable column names for appropriate conversion to `GRanges`.
`keep.extra.columns`	Keep extra columns parameter from `GenomicRanges::makeGRangesFromDataFrame()`.

Details

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

Value

A `GRanges` that may or may not `keep.extra.columns`, and that may or may not be stranded, depending on whether there is strand column in the `dframe`.

Examples

```
# Example with just chr, start, end
peaks_df = data.frame(
chr = c('chr1','chr2','chr3'),
start = c(35,74,235),
end = c(46,83,421),
stringsAsFactors = FALSE)
peaks = load_peaks(peaks_df)

# Example with extra columns
peaks_df = data.frame(
chr = c('chr1','chr2','chr3'),
start = c(35,74,235),
end = c(46,83,421),
strand = c('+','-','+'),
score = c(36, 747, 13),
stringsAsFactors = FALSE)
peaks = load_peaks(peaks_df, keep.extra.columns = TRUE)
```

---

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

Description

For each gene_id, 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 gene_id, 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_chipenrich_spline` *QC plot for ChIP-Enrich*

Description

A plot showing an approximation of the empirical spline used to model the relationship between a gene having a peak and the locus length. For visual clarity, genes are binned into groups of 25 after sorting by locus length. Expected fits assuming independence of locus length and presence of a peak, and assuming proportionality of locus length and presence of a peak are given to demonstrate deviation from either for the dataset.

Usage

```
plot_chipenrich_spline(peaks, locusdef = "nearest_tss",
  genome = supported_genomes(), mappability = NULL, legend = TRUE,
  xlim = NULL)
```

Arguments

`peaks`	Either a file path or a `data.frame` of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a `data.frame` A BEDX+Y style `data.frame`. See `GenomicRanges::makeGRangesFromDataFrame` for acceptable column names.
`locusdef`	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or `supported_locusdefs`. Alternately, a file path for a custom locus definition. NOTE: Must be for a `supported_genome()`, and must have columns

	'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
genome	One of the supported_genomes().
mappability	One of NULL, a file path to a custom mappability file, or an integer for a valid read length given by supported_read_lengths. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.
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')

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

---

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 = supported_genomes())
```

Arguments

peaks	Either a file path or a data.frame of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a data.frame A BEDX+Y style data.frame. See GenomicRanges::makeGRangesFromDataFrame for acceptable column names.
genome	One of the supported_genomes().

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, genome = 'hg19')
```

---

plot_gene_coverage *QC plot for Broad-Enrich*

Description

Create a plot showing the relationship between locus length and the proportion of gene loci covered by peaks.

Usage

```
plot_gene_coverage(peaks, locusdef = "nearest_tss",
  genome = supported_genomes(), mappability = NULL, legend = TRUE,
  xlim = NULL)
```

Arguments

peaks	Either a file path or a `data.frame` of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a `data.frame` A BEDX+Y style `data.frame`. See `GenomicRanges::makeGRangesFromDataFrame` for acceptable column names.
locusdef	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or `supported_locusdefs`. Alternately, a file path for a custom locus definition. NOTE: Must be for a `supported_genome()`, and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
genome	One of the `supported_genomes()`.
mappability	One of `NULL`, a file path to a custom mappability file, or an `integer` for a valid read length given by `supported_read_lengths`. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.
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_H3K4me3_GM12878, package = 'chipenrich.data')

# Create the plot for a different locus definition
# to compare the effect.
plot_gene_coverage(peaks_H3K4me3_GM12878, locusdef = 'nearest_gene', genome = 'hg19')
```

`plot_polyenrich_spline` *QC plot for Poly-Enrich*

Description

Create a plot the relationship between number of peaks assigned to a gene and locus length. The plot shows an empirical fit to the data using a binomial smoothing spline.

Usage

```
plot_polyenrich_spline(peaks, locusdef = "nearest_tss",
  genome = supported_genomes(), mappability = NULL, legend = TRUE,
  xlim = NULL, ylim = NULL)
```

Arguments

`peaks`	Either a file path or a `data.frame` of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a `data.frame` A BEDX+Y style `data.frame`. See `GenomicRanges::makeGRangesFromDataFrame` for acceptable column names.
`locusdef`	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or `supported_locusdefs`. Alternately, a file path for a custom locus definition. NOTE: Must be for a `supported_genome()`, and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
`genome`	One of the `supported_genomes()`.
`mappability`	One of `NULL`, a file path to a custom mappability file, or an `integer` for a valid read length given by `supported_read_lengths`. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.

legend	If true, a legend will be drawn on the plot.
xlim	Set the x-axis limit. NULL means select x-lim automatically.
ylim	Set the y-axis limit. NULL means select y-lim automatically.

Value

A trellis plot object.

Examples

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

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

---

polyenrich *Run Poly-Enrich on narrow genomic regions*

Description

Poly-Enrich is designed for narrow peaks, but where there are 100,000s of peaks which results in nearly every gene locus containing a peak. For example, ChIP-seq experiments for transcription factors. For more details, see the 'Poly-Enrich Method' section below. For help choosing a method, see the 'Choosing A Method' section below, or see the vignette.

Usage

```
polyenrich(peaks, out_name = "polyenrich", out_path = getwd(),
  genome = supported_genomes(), genesets = c("GOBP", "GOCC", "GOMF"),
  locusdef = "nearest_tss", method = "polyenrich", weighting = NULL,
  mappability = NULL, qc_plots = TRUE, min_geneset_size = 15,
  max_geneset_size = 2000, randomization = NULL, n_cores = 1)
```

Arguments

peaks	Either a file path or a data.frame of peaks in BED-like format. If a file path, the following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to 'chr', 'start', and 'end' and that there is either no header column, or it is commented out. If a data.frame A BEDX+Y style data.frame. See GenomicRanges::makeGRangesFromDataFrame for acceptable column names.
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 "polyenrich", and a file "polyenrich_results.tab" is produced. If qc_plots is set, then a file "polyenrich_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 polyenrich.

out_path	Directory to which results files will be written out. Defaults to the current working directory as returned by getwd.
genome	One of the supported_genomes().
genesets	A character vector of geneset databases to be tested for enrichment. See supported_genesets(). Alternately, a file path to a a tab-delimited text file with header and first column being the geneset ID or name, and the second column being Entrez Gene IDs. For an example custom gene set file, see the vignette.
locusdef	One of: 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. For a description of each, see the vignette or supported_locusdefs. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id' or 'geneid'. For an example custom locus definition file, see the vignette.
method	A character string specifying the method to use for enrichment testing. Current options are polyenrich and polyenrich_weighted.
weighting	A character string specifying the weighting method if method is chosen to be 'polyenrich_weighted'. Current options are: 'signalValue', 'logsignalValue', and 'multiAssign'.
mappability	One of NULL, a file path to a custom mappability file, or an integer for a valid read length given by supported_read_lengths. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. For an example custom mappability file, see the vignette. Default value is NULL.
qc_plots	A logical variable that enables the automatic generation of plots for quality control.
min_geneset_size	Sets the minimum number of genes a gene set may have to be considered for enrichment testing.
max_geneset_size	Sets the maximum number of genes a gene set may have to be considered for enrichment testing.
randomization	One of NULL, 'complete', 'bylength', or 'bylocation'. See the Randomizations section below.
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:

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

chr 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.

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

gene_symbol is the official gene symbol for the gene_id (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 is the closest TSS to this peak (for any gene, not necessarily the gene this peak was assigned to.)

nearest_tss_gene is the gene having the closest TSS to the peak (should be the same as gene_id when using the nearest TSS locus definition.)

nearest_tss_gene_strand is the strand of the gene with the closest TSS.

`peaks_per_gene`

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

gene_id 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 has a peak.

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

Poly-Enrich Method

The Poly-Enrich method uses the number of peaks in genes in its model for enrichment: `num_peaks ~ GO + s(log10_le`
Here, `GO` is a binary vector indicating whether a gene is in the gene set being tested, `num_peaks` is a numeric vector indicating the number of peaks in each gene, and `s(log10_length)` is a negative binomial cubic smoothing spline which adjusts for the relationship between the number of peaks in a gene and locus length.

Poly-Enrich Weighting Options

Poly-Enrich also allows weighting of individual peaks. Currently the options are:

'signalValue:' weighs each peak based on the Signal Value given in the narrowPeak format or a user-supplied column, normalized to have mean 1.

'logsignalValue:' weighs each peak based on the log Signal Value given in the narrowPeak format or a user-supplied column, normalized to have mean 1.

'multiAssign:' weighs each peak by the inverse of the number of genes it is assigned to.

Choosing A Method

The following guidelines are intended to help select an enrichment function:

broadenrich(): is designed for use with broad peaks that may intersect multiple gene loci, and cumulatively cover greater than 5% of the genome. For example, ChIP-seq experiments for histone modifications.

chipenrich(): is designed for use with 1,000s or 10,000s of narrow peaks which results in fewer gene loci containing a peak overall. For example, ChIP-seq experiments for transcription factors.

polyenrich(): is also designed for narrow peaks, but where there are 100,000s of peaks which results in nearly every gene locus containing a peak. For example, ChIP-seq experiments for transcription factors. Generally works better for experiments with more than 40,000 peaks.

Randomizations

Randomization of locus definitions allows for the assessment of Type I Error under the null hypothesis. The randomization codes are:

`NULL`**:** No randomizations, the default.

'complete': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together, without regard for the chromosome location, or locus length. The null hypothesis is that there is no true gene set enrichment.

'bylength': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together within bins of 100 genes sorted by locus length. The null hypothesis is that there is no true gene set enrichment, but with preserved locus length relationship.

'bylocation': Shuffle the `gene_id` and `symbol` columns of the `locusdef` together within bins of 50 genes sorted by genomic location. The null hypothesis is that there is no true gene set enrichment, but with preserved genomic location.

The return value with a selected randomization is the same list as without. To assess the Type I error, the `alpha` level for the particular data set can be calculated by dividing the total number of gene sets with p-value < `alpha` by the total number of tests. Users may want to perform multiple randomizations for a set of peaks and take the median of the `alpha` values.

See Also

Other enrichment functions: broadenrich, chipenrich

Examples

```
# Run Poly-Enrich using an example dataset, assigning peaks to the nearest TSS,
# and on a small custom geneset
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 = polyenrich(peaks_E2F4, method='polyenrich', locusdef='nearest_tss',
genome = 'hg19', 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
```

---

postprocess_peak_grs *A helper function to post-process peak GRanges*

Description

Check for overlapping input regions, sort peaks, and force peak names

Usage

```
postprocess_peak_grs(gr)
```

Arguments

gr	A GRanges of input peaks.

Value

A GRanges that is sorted if the seqinfo is set, and has named peaks.

---

post_process_enrichments *Post process the* data.frame *of enrichment results*

Description

Post process the data.frame of enrichment results

Usage

```
post_process_enrichments(enrich)
```

Arguments

enrich	A data.frame of the enrichment results from broadenrich(), chipenrich(), or polyenrich() created by rbinding the list of enrichment results for each of the genesets.

Value

A reformatted data.frame with columns in a specific order, filtered of enrichment tests that failed, and ordered first by enrichment 'Status' (if present) and then 'P.value'.

---

read_bed *Read files containing peaks or genomic regions*

Description

The following formats are fully supported via their file extensions: .bed, .broadPeak, .narrowPeak, .gff3, .gff2, .gff, and .bedGraph or .bdg. BED3 through BED6 files are supported under the .bed extension. Files without these extensions are supported under the conditions that the first 3 columns correspond to chr, start, and end and that there is either no header column, or it is commented out. Files may be compressed with gzip, and so might end in .narrowPeak.gz, for example. For files with extension support, the rtracklayer::import() function is used to read peaks, so adherence to the mentioned file formats is necessary.

Usage

```
read_bed(file_path)
```

Arguments

file_path	A path to a file with input peaks/regions. See extended description above for details about file support.

Details

NOTE: Header rows must be commented with # to be ignored. Otherwise, an error may result.

NOTE: A warning is given if any input regions overlap. In the case of enrichment testing with method = 'broadenrich', regions should be disjoint.

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

Value

A GRanges with mcols matching any extra columns.

Examples

```
# Example of generic .txt file with peaks
file = system.file('extdata', 'test_header.txt', package = 'chipenrich')
peaks = read_bed(file)

# Example of BED3
```

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

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

# Example of gzipped broadPeak
file = system.file('extdata', 'test.broadPeak.gz', package = 'chipenrich')
peaks = read_bed(file)

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

`read_geneset` *Function to read custom gene sets from file*

Description

This function reads a two-columned tab-delimited text file (with header). Column names are ignored, but the first column should be geneset names or IDs and the second column should be Entrez Gene IDs.

Usage

```
read_geneset(file_path)
```

Arguments

`file_path`	A file path for the custom gene set.

Value

A `GeneSet` class object.

`read_ldef` *Function to read custom locus definition from file*

Description

This function reads a tab-delimited text (with a header) file that should have columns 'chr', 'start', 'end', and a column named 'gene_id' (or 'geneid') with the Entrez Gene ID. If a `supported_genomes()` is given, then a column of gene symbols named 'symbol', will be added. If an unsupported genome is used there are two options: 1) Have a column named 'symbols' with the gene symbols in the custom locus definition, and leave `genome = NA`, or 2) leave `genome = NA`, do not provide gene symbols, and NAs will be used.

Usage

```
read_ldef(file_path, genome = NA)
```

Arguments

file_path	A file path for the custom locus definition.
genome	A genome from supported_genomes(), default NA.

Value

A LocusDefinition class object with slots dframe, granges, genome.build, and organism.

---

read_mappa *Function to read custom mappability files*

Description

This function reads a two-columned tab-delimited text file (with header). Expected column names are 'mappa' and 'gene_id'. Each line is for a unique 'gene_id' and contains the mappability (between 0 and 1) for that gene.

Usage

```
read_mappa(file_path)
```

Arguments

file_path	A file path for the custom mappability.

Value

A data.frame containing gene_id and mappa columns.

---

recode_peaks *Recode a vector of number of peaks to binary based on threshold*

Description

Recode a vector of number of peaks to binary based on threshold

Usage

```
recode_peaks(num_peaks, threshold = 1)
```

Arguments

num_peaks	An integer vector representing numbers of peaks per gene.
threshold	An integer specifying the minimum number of peaks required to code as 1.

Value

An binary vector where an entry is 1 if the corresponding entry of num_peaks is >= threshold and is otherwise 0.

reset_ncores_for_windows *Reset n_cores for Windows*

Description

We use `parallel::mclapply` for multicore geneset enrichment testing, but this function supports more than one core if the OS is not Windows. If the OS is windows, the number of cores (`mc.cores`) must be set to 1.

Usage

```
reset_ncores_for_windows(n_cores)
```

Arguments

`n_cores`	An `integer` passed to `broadenrich()`, `chipenrich()`, or `polyenrich()` indicating the number of cores to use for enrichment testing.

Value

Either the original `n_cores` if the OS is not Windows, or 1 if the OS is Windows.

setup_genesets *Function to setup genesets*

Description

Function to setup genesets

Usage

```
setup_genesets(gs_codes, ldef_obj, genome, min_geneset_size, max_geneset_size)
```

Arguments

`gs_codes`	A character vector of geneset databases to be tested for enrichment. See `supported_genesets()`. Alternately, a file path to a a tab-delimited text file with header and first column being the geneset ID or name, and the second column being Entrez Gene IDs.
`ldef_obj`	A `LocusDefinition` object to use for filtering gene sets based on which genes are defined in the locus defintion.
`genome`	One of the `supported_genomes()`.
`min_geneset_size`	Sets the minimum number of genes a gene set may have to be considered for enrichment testing.
`max_geneset_size`	Sets the maximum number of genes a gene set may have to be considered for enrichment testing.

Value

A list with components consisting of GeneSet objects for each of the elements of genesets. NOTE: Custom genesets must be run separately from built in gene sets.

---

setup_locusdef *Function to setup locus definitions*

Description

Function to setup locus definitions

Usage

```
setup_locusdef(ldef_code, genome, randomization = NULL)
```

Arguments

ldef_code	One of 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id', or 'geneid'.
genome	One of the supported_genomes().
randomization	One of NULL, 'complete', 'bylength', or 'bylocation'. See the Randomizations section in ?chipenrich. Default NULL.

Value

A list with components ldef and tss.

---

setup_mappa *Function to setup mappability*

Description

Function to setup mappability

Usage

```
setup_mappa(mappa_code, genome, ldef_code, ldef_obj)
```

Arguments

mappa_code	One of NULL, a file path to a custom mappability file, or an integer for a valid read length given by supported_read_lengths. If a file, it should contain a header with two column named 'gene_id' and 'mappa'. Gene IDs should be Entrez IDs, and mappability values should range from 0 and 1. Default value is NULL.
genome	One of the supported_genomes().
ldef_code	One of 'nearest_tss', 'nearest_gene', 'exon', 'intron', '1kb', '1kb_outside', '1kb_outside_upstream', '5kb', '5kb_outside', '5kb_outside_upstream', '10kb', '10kb_outside', '10kb_outside_upstream'. Alternately, a file path for a custom locus definition. NOTE: Must be for a supported_genome(), and must have columns 'chr', 'start', 'end', and 'gene_id', or 'geneid'.
ldef_obj	A LocusDefinition object.

Value

A data.frame with columns gene_id and mappa.

---

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

The locus definitions are defined as below. For advice on selecting a locus definition, see the 'Selecting A Locus Definition' section below.

nearest_tss: The locus is the region spanning the midpoints between the TSSs of adjacent genes.

nearest_gene: The locus is the region spanning the midpoints between the boundaries of each gene, where a gene is defined as the region between the furthest upstream TSS and furthest downstream TES for that gene. If two gene loci overlap each other, we take the midpoint of the overlap as the boundary between the two loci. When a gene locus is completely nested within another, we create a disjoint set of 3 intervals, where the outermost gene is separated into 2 intervals broken apart at the endpoints of the nested gene.

1kb: The locus is the region within 1kb of any of the TSSs belonging to a gene. If TSSs from two adjacent genes are within 2 kb of each other, we use the midpoint between the two TSSs as the boundary for the locus for each gene.

1kb_outside_upstream: The locus is the region more than 1kb upstream from a TSS to the midpoint between the adjacent TSS.

1kb_outside: The locus is the region more than 1kb upstream or downstream from a TSS to the midpoint between the adjacent TSS.

5kb: The locus is the region within 5kb of any of the TSSs belonging to a gene. If TSSs from two adjacent genes are within 10 kb of each other, we use the midpoint between the two TSSs as the boundary for the locus for each gene.

5kb_outside_upstream: The locus is the region more than 5kb upstream from a TSS to the midpoint between the adjacent TSS.

5kb_outside: The locus is the region more than 5kb upstream or downstream from a TSS to the midpoint between the adjacent TSS.

10kb: The locus is the region within 10kb of any of the TSSs belonging to a gene. If TSSs from two adjacent genes are within 20 kb of each other, we use the midpoint between the two TSSs as the boundary for the locus for each gene.

10kb_outside_upstream: The locus is the region more than 10kb upstream from a TSS to the midpoint between the adjacent TSS.

10kb_outside: The locus is the region more than 10kb upstream or downstream from a TSS to the midpoint between the adjacent TSS.

exon: Each gene has multiple loci corresponding to its exons. Overlaps between different genes are allowed.

intron: Each gene has multiple loci corresponding to its introns. Overlaps between different genes are allowed.

Usage

```
supported_locusdefs()
```

Value

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

Selecting A Locus Definition

For a transcription factor ChIP-seq experiment, selecting a particular locus definition for use in enrichment testing can have implications relating to how the TF regulates genes. For example, selecting the '1kb' locus definition will imply that the biological processes found enriched are a result of TF regulation near the promoter. In contrast, selecting the '5kb_outside' locus definition will imply that the biological processes found enriched are a result of TF regulation distal from the promoter.

Selecting a locus definition can also help reduce the noise in the enrichment tests. For example, if a TF is known to primarily regulate genes by binding around the promoter, then selecting the '1kb' locus definition can help to reduce the noise from TSS-distal peaks in the enrichment testing.

The `plot_dist_to_tss` QC plot displays where genomic regions fall relative to TSSs genome-wide, and can help inform the choice of locus definition. For example, if many peaks fall far from the TSS, the 'nearest_tss' locus definition may be a good choice because it will capture all input genomic regions, whereas the '1kb' locus definition may not capture many of the input genomic regions and adversely affect the enrichment testing.

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