Package 'chipenrich'

October 7, 2014

Type Package

Title Gene set enrichment for ChIP-seq peak data

Version 1.2.0

Date 2013-08-31

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.

License GPL-3

Imports chipenrich.data, methods, GenomicRanges (>= 1.10.0), IRanges (>= 1.16.0), mgcv, plyr (>= 1.7.0), lattice, latticeExtra,grid, stringr (>= 0.6), reshape

Depends R (>= 2.15.1)

LazyLoad yes

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

Maintainer Ryan P. Welch <welchr@umich.edu>

biocViews Software, GeneSetEnrichment

R topics documented:

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

Description

Run gene set enrichment testing (ChIP-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.

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)
```

Arguments

`peaks`	Either a data frame, or file containing peaks, or a BED file. The data frame should have 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX". The file should be tab-delimited and contain a header with at least the columns chrom, start, and end.
`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 string denoting the gene set enrichment test to be performed (ChIP-Enrich ('chipenrich'), 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'.

`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.

`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.)

`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.

`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.

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

Details

The `chipenrich` function is used to run gene set enrichment tests on a file or data frame contaning peaks called from a ChIP-seq experiment. It can currently run the following tests:

- ChIP-Enrich (`method=chipenrich`) is the main method, which is similar to Fisher's exact test in that it is a test of association between genes having a peak, and genes belonging to a predefined gene set, but that also adjusts for gene locus length and sequence mappability (optional). The method uses binomial smoothing splines in a generalized additive model framework, see (Wood, 2010) or the `gam` function for more details.
- Fisher's exact test (`method=fet`) is the naive approach, where peaks are assigned to a gene using the locus definitions set using the `locusdef` option, and then each set of genes is tested using a 2x2 table tabulating whether genes have a peak, and whether they belong to the set of genes being tested. Note that this method does not adjust for gene locus length.

The `peaks` option should be either:

- A data frame with 3 columns named: chrom, start, and end; denoting the chromosome, starting position, and ending position of the peak. Chromosome should be in UCSC format, e.g. chrX, chrY, chr22, etc.
- Character vector representing the path to a file containing the data frame outlined above. The file must be tab-delimited and the header must exist. Additional columns can exist, so long as they do not contain tab characters.
- A BED file, following UCSC's formatting. The file must have a .bed extension to be read as a BED file.

The `locusdef` option controls how peaks are assigned to genes. A number of options are available and can be listed using `supported_locusdefs`. We currently support:

- `nearest_tss`: assigns peaks to the gene with the closest transcription start site (TSS). Each gene locus stretches from the midpoint upstream between TSSs, and the midpoint downstream between TSSs.
- `nearest_gene`: assigns peaks to the nearest gene. Each gene locus stretches from the upstream midpoint between genes and the downstream midpoint between genes.
- `1kb`: each gene locus is defined as the region 1KB up- and down-stream of the TSS.
- `5kb`: each gene locus is defined as the region 5KB up- and down-stream of the TSS.

The `use_mappability` option enables correction for sequence mappability. This is done by multiplying the gene locus length by the mappability of the locus to compute the mappable genome length. The log10 mappable locus length is then used in the model rather than the log10 locus length. See the vignette for a complete description of mappability.

If `use_mappability` is enabled, the user should specify `read_length`, which should roughly correspond to the length of sequencing reads from the ChIP-seq experiment. Mappability of sequence is calculated using "reads" of a given length. See the vignette for a description of how mappability is calculated given a gene locus and read length. A list of supported read lengths can be found by using the `supported_read_lengths` function.

The user can also provide their own per-gene mappability scores using the `mappa_file` option. This overrides both `use_mappability` and `read_length` options above. The file should contain two columns: geneid, and mappa. Gene ID should be a valid Entrez gene ID, and mappa is a mappability score between 0 and 1.

The `qc_plots` option enables the automatic generation of quality control plots aimed at giving the user a better understanding of their data. Currently three plots are created:

- A spline fit plot showing the relationship between the locus length of a gene and the likelihood of that gene having a peak given the data. See `plot_spline_length` to generate the plot separately, and for a description of the plot features.
- A histogram of the distance from each peak to the nearest transcription start site (TSS) of any gene. See `plot_dist_to_tss` to generate the plot separately, and for a description of the plot features.

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:

chrom is the chromosome the peak originated from.

peak_start is starting 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 starting 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 geneid when using the nearest TSS locus definition.)

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

`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.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`.

Author(s)

Ryan Welch <welchr@umich.edu>

References

Welch RP*, Lee C*, Smith RA, Imbriano P, Scott LJ, Sartor MA. ChIP-Enrich: Gene set enrichment testing for ChIP-Seq data (in preparation.)

Wood, S. mgcv: GAMs with GCV/AIC/REML smoothness estimation and GAMMs by PQL. R package version, 1.6-2 (2010).

See Also

For lists of supported arguments to chipenrich:

- supported_genomes for a list of available genomes.
- supported_genesets for a list of available geneset databases (KEGG, GO, etc.)
- supported_locusdefs for a list of available gene locus definitions.
- supported_methods for a list of available enrichment testing methods.

For making various QC plots from your peak data:

- plot_spline_length
- plot_dist_to_tss

Examples

```
library(chipenrich.data)
library(chipenrich)

# Run ChipEnrich using an example dataset, assigning peaks to the nearest TSS,
# testing all Biocarta and Panther pathways
data(peaks_E2F4)
results = chipenrich(peaks_E2F4,method=chipenrich,locusdef=nearest_tss,
genesets=c(biocarta_pathway,panther_pathway),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
```

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	Either a data frame, or file containing peaks, or a BED file. The data frame should have 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX". The file should be tab-delimited and contain a header with at least the columns chrom, start, and end.
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.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
library(chipenrich.data)
library(chipenrich)

# Create histogram of distance from peaks to nearest TSS.
data(peaks_E2F4)
plot_dist_to_tss(peaks_E2F4)
```

`plot_expected_peaks` *Plot the expected number of peaks given locus length for each gene.*

Description

Create a plot showing the expected count of peaks per gene given their locus lengths.

Also plotted are confidence intervals for the expected count, and the actual observed number of peaks per gene.

Usage

```
plot_expected_peaks(peaks,
  locusdef = "nearest_tss",
  genome = "hg19",
  use_mappability = F,
  read_length = 36,
  mappa_file = NULL
)
```

Arguments

`peaks`	Either a data frame, or file containing peaks, or a BED file. The data frame should have 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX". The file should be tab-delimited and contain a header with at least the columns chrom, start, and end.
`locusdef`	A string denoting the locus definition to be used. A locus definition controls how peaks are assigned to genes. See `supported_locusdefs` for a list of supported definitions.
`genome`	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the `supported_genomes` function.
`use_mappability`	If true, each gene's locus length is corrected for by mappability.
`read_length`	If using mappability (see above), the read length should match the length of reads used in the original experiment.
`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.

Details

The x-axis is gene locus length (for the defined `locusdef`.) The y-axis is count of peaks. Each blue dot represents the observed count of peaks assigned to a gene.

The black line represents the expected number of peaks given locus length.

Also drawn are the 5 and 95% percentiles of a Poisson distribution for the expected number of peaks, and the 5 and 95% percentiles adjusted for the number of genes (Bonferroni adjustment - e.g. 0.05 / # of genes.)

Value

A trellis plot object.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
library(chipenrich.data)
library(chipenrich)

# Expected peak count plot for the E2F4 dataset.
data(peaks_E2F4)
plot_expected_peaks(peaks_E2F4,genome=hg19)

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


# Create the plot, but write the result to a PDF
# instead of displaying it interactively.
pdf("expected_peak_plot.pdf");
p = plot_expected_peaks(peaks_E2F4,genome=hg19);
dev.off();
```

---

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	Either a data frame, or file containing peaks, or a BED file. The data frame should have 3 columns: chrom, start, and end. Chrom should follow UCSC convention, e.g. "chrX". The file should be tab-delimited and contain a header with at least the columns chrom, start, and end.
locusdef	A string denoting the locus definition to be used. A locus definition controls how peaks are assigned to genes. See supported_locusdefs for a list of supported definitions.
genome	A string indicating the genome upon which the peaks file is based. Supported genomes are listed by the supported_genomes function.
use_mappability	If true, each gene's locus length is corrected for by mappability.
read_length	If using mappability (see above), the read length should match the length of reads used in 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.

Details

The x-axis represents the log10 of the gene locus length (defined by the locus definition.) The y-axis is the probability of a gene being assigned a peak. Each black dot on the plot is a bin of 25 genes, the y-axis value is then the proportion of peaks that were assigned to genes in that bin, and the x-axis value is the average of the locus lengths of genes in that bin.

The spline curve is fit using a binomial smoothing spline model, see chipenrich for more information. This curve is created by modeling presence of peak (a 0/1 binary variable denoting whether the gene was assigned a peak) given the gene locus length.

The random genes curve represents the model where each gene has the same probability of being assigned a peak given the total number of peaks in the dataset.

The "random peaks across genome" curve represents the model where the probability of a gene being assigned at least 1 peak is dependent on its length:

$$p(peak|L_{locus}) = 1 - (1 - \frac{L_{locus}}{L_{genome}})^{n_{peaks}}$$

L_{locus} is the length of the gene locus, L_{genome} is the length of the sampleable genome, and n_{peaks} is the number of peaks in the dataset.

Value

A trellis plot object.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
library(chipenrich.data)
library(chipenrich)

# Spline plot for E2F4 example peak dataset.
data(peaks_E2F4)
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);


# Create the plot, but write the result to a PDF instead of displaying it interactively.
pdf("spline_plot.pdf");
p = plot_spline_length(peaks_E2F4,genome=hg19);
dev.off();
```

`supported_genesets` *List supported genesets*

Description

Obtain list of supported genesets

Usage

```
supported_genesets()
```

Value

A character vector listing the currently supported genesets by the `chipenrich` function.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

`chipenrich`

Examples

```
supported_genesets()
```

supported_genomes *List supported genomes*

Description

Obtain list of supported genomes

Usage

```
supported_genomes()
```

Value

A character vector listing the currently supported genomes by the chipenrich function.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
supported_genomes()
```

supported_locusdefs *List supported locus definitions*

Description

Obtain list of supported locus definitions for assigning peaks to genes

Usage

```
supported_locusdefs()
```

Value

A character vector listing the currently supported locus definitions by the chipenrich function.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
supported_locusdefs()
```

`supported_methods` *List supported methods (enrichment tests)*

Description

Obtain list of supported methods (enrichment tests)

Usage

```
supported_methods()
```

Value

A character vector listing the currently supported methods by the chipenrich function.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
supported_methods()
```

supported_read_lengths *List supported read lengths*

Description

Obtain list of supported read lengths (used when adjusting for mappability)

Usage

```
supported_read_lengths()
```

Value

A character vector listing the currently supported mappability read lengths by the chipenrich function.

Author(s)

Ryan Welch <welchr@umich.edu>

See Also

chipenrich

Examples

```
supported_read_lengths()
```

Index