Package ‘annotatr’

May 15, 2024

Title Annotation of Genomic Regions to Genomic Annotations

Version 1.30.0

Date 2021-11-20

Description Given a set of genomic sites/regions (e.g. ChIP-seq peaks, CpGs, differentially methylated CpGs or regions, SNPs, etc.) it is often of interest to investigate the intersecting genomic annotations. Such annotations include those relating to gene models (promoters, 5'UTRs, exons, introns, and 3'UTRs), CpGs (CpG islands, CpG shores, CpG shelves), or regulatory sequences such as enhancers. The annotatr package provides an easy way to summarize and visualize the intersection of genomic sites/regions with genomic annotations.

Depends R (>= 3.4.0)

Imports AnnotationDbi, AnnotationHub, dplyr, GenomicFeatures, GenomicRanges, GenomeInfoDb (>= 1.10.3), ggplot2, IRanges, methods, readr, regioneR, reshape2, rtracklayer, S4Vectors (>= 0.23.10), stats, utils

Suggests BiocStyle, devtools, knitr, org.Dm.eg.db, org.Gg.eg.db, org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db, rmarkdown, roxygen2, testthat, TxDb.Dmelanogaster.UCSC.dm3.ensGene, TxDb.Dmelanogaster.UCSC.dm6.ensGene, TxDb.Ggallus.UCSC.galGal5.refGene, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Mmusculus.UCSC.mm9.knownGene, TxDb.Mmusculus.UCSC.mm10.knownGene, TxDb.Rnorvegicus.UCSC.rn4.ensGene, TxDb.Rnorvegicus.UCSC.rn5.refGene, TxDb.Rnorvegicus.UCSC.rn6.refGene

VignetteBuilder knitr

BugReports https://www.github.com/rcavalcante/annotatr/issues

License GPL-3

LazyData true

RoxygenNote 7.1.2

biocViews Software, Annotation, GenomeAnnotation, FunctionalGenomics, Visualization

git_url https://git.bioconductor.org/packages/annotatr

git_branch RELEASE_3_19

git_last_commit aa1c8ba

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-15

Author Raymond G. Cavalcante [aut, cre],
Maureen A. Sartor [ths]

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

Contents

`annotate_regions` *A function to intersect user region data with annotation data*

Description

Annotate genomic regions to selected genomic annotations while preserving the data associated with the genomic regions.

Usage

```
annotate_regions(
  regions,
  annotations,
  minoverlap = 1L,
  ignore.strand = TRUE,
  quiet = FALSE
)
```

Arguments

`regions`	The GRanges object returned by `read_regions()`.
`annotations`	A character vector of annotations to build. Valid annotation codes are listed with `builtin_annotations()`. The "basicgenes" shortcut builds the following regions: 1-5Kb upstream of TSSs, promoters, 5UTRs, exons, introns, and 3UTRs. The "cpgs" shortcut builds the following regions: CpG islands, shores, shelves, and interCGI regions. NOTE: Shortcuts need to be appended by the genome, e.g. `hg19_basicgenes`. Custom annotations whose names are of the form `[genome]_custom_[name]` should also be included. Custom annotations should be read in and converted to `GRanges` with `read_annotations()`. They can be for a `supported_genome()`, or for an unsupported genome.
`minoverlap`	A scalar, positive integer, indicating the minimum required overlap of regions with annotations.
`ignore.strand`	Logical indicating whether strandedness should be respected in findOverlaps(). Default FALSE.
`quiet`	Print progress messages (FALSE) or not (TRUE).

Value

A `GRanges` where the `granges` are from the regions, and the `mcols` include the `mcols` from the regions and a column with the annotation `GRanges`.

Examples

```
r_file = system.file('extdata', 'test_read_multiple_data_nohead.bed', package='annotatr')
extraCols = c(pval = 'numeric', mu1 = 'integer', mu0 = 'integer', diff_exp = 'character')
 r = read_regions(con = r_file, extraCols = extraCols, rename_score = 'coverage')

 # Get premade CpG annotations
 data('annotations', package = 'annotatr')

 a = annotate_regions(
     regions = r,
     annotations = annotations,
     ignore.strand = TRUE)
```

`annotations` *example_annotations data*

Description

A `GRanges` of precomputed annotations for CpG features. Created by doing `build_annotations(genome='hg19', annotations = 'hg19_cpgs')`.

Usage

```
annotations
```

Format

A `GRanges` object with the CpG feature annotations for hg19 and containing `mcols`:

id The internal ID for the annotation

tx_id All NA, since these are not associated with tx_ids

gene_id All NA, since there are not associated Entrez IDs

symbols All NA, since there are not associated gene symbols

type A character indicating the type of annotation. Including: 'hg19_cpg_islands', 'hg19_cpg_shores', 'hg19_cpg_shelves', and 'hg19_cpg_inter'.

Source

The AnnotationHub resource for hg19 CpG features.

annotatr *annotatr: Annotation of Genomic Regions to Functional Annotations*

Description

Given a set of genomic sites/regions (e.g. ChIP-seq peaks, CpGs, differentially methylated CpGs or regions, SNPs, etc.) it is often of interest to investigate the intersecting functional annotations. Such annotations include those relating to gene models (promoters, 5'UTRs, exons, introns, and 3'UTRs), CpGs (CpG islands, CpG shores, CpG shelves), the non-coding genome, and enhancers. The annotatr package provides an easy way to summarize and visualize the intersection of genomic sites/regions with the above functional annotations.

annotatr_cache *A global-variable to hold custom annotations loaded in an R session*

Description

Code thanks to Martin Morgan. This is a global variable that will store custom annotations that a user reads in during a session in which annotatr is loaded.

Usage

```
annotatr_cache
```

Format

An object of class `list` of length 3.

Value

An environment to contain custom annotations from `read_annotations`.

Examples

```
# Example usage
annotatr_cache$set("foo", 1:10)
annotatr_cache$get("foo")

# Read in a BED3 file as a custom annotation
file = system.file('extdata', 'test_annotations_3.bed', package='annotatr')
# The custom annotation is added to the annotatr_cache environment in this function
read_annotations(con = file, name = 'test', genome = 'hg19')
# The result of read_annotations() is not visible in .GlobalEnv, instead
# need to use the get method
print(annotatr_cache$get('hg19_custom_test'))
# See what is in the annotatr_cache
annotatr_cache$list_env()
```

build_ah_annots — *A helper function to build arbitrary annotatinos from AnnotationHub*

Description

A helper function to build arbitrary annotatinos from AnnotationHub

Usage

```
build_ah_annots(genome, ah_codes, annotation_class)
```

Arguments

genome	The genome assembly.
ah_codes	A named character vector giving the AnnotationHub accession number (e.g. AH23256), and whose name describes what the annotation is (e.g. Gm12878_H3K4me3).
annotation_class	A string to name the group of annotations in ah_codes

Value

A GRanges object stored in annotatr_cache. To view an annotation built with this function, do annotatr_cache$get(name). To add these annotations to a set of annotations, include '[genome]_[annotation_class]_[in the call to build_annotations(). See example below.

Examples

```
# Create a named vector for the AnnotationHub accession codes with desired names
h3k4me3_code = c('Gm12878' = 'AH23256')
# Fetch ah_codes from AnnotationHub and create annotations annotatr understands
build_ah_annots(genome = 'hg19', ah_codes = h3k4me3_code, annotation_class = 'H3K4me3')
# The annotations as they appear in annotatr_cache
annot_name = c('hg19_H3K4me3_Gm12878')
# Build the annotations right before annotating any regions
annotations = build_annotations(genome = 'hg19', annotations = annot_name)
```

build_annotations — *A function to build annotations from TxDb.* and AnnotationHub resources*

Description

Create a GRanges object consisting of all the desired annotations. Supported annotation codes are listed by builtin_annotations(). The basis for enhancer annotations are FANTOM5 data, the basis for CpG related annotations are CpG island tracks from AnnotationHub, and the basis for genic annotations are from the TxDb.* and org.db group of packages.

Usage

```
build_annotations(genome, annotations)
```

Arguments

genome	The genome assembly.
annotations	A character vector of annotations to build. Valid annotation codes are listed with `builtin_annotations()`. The "basicgenes" shortcut builds the following regions: 1-5Kb upstream of TSSs, promoters, 5UTRs, exons, introns, and 3UTRs. The "cpgs" shortcut builds the following regions: CpG islands, shores, shelves, and interCGI regions. NOTE: Shortcuts need to be appended by the genome, e.g. `hg19_basicgenes`. Custom annotations whose names are of the form `[genome]_custom_[name]` should also be included. Custom annotations should be read in and converted to `GRanges` with `read_annotations()`. They can be for a `supported_genome()`, or for an unsupported genome.

Value

A `GRanges` object of all the `annotations` combined. The `mcols` are `id, tx_id, gene_id, symbol, type`. The `id` column is a unique name, the `tx_id` column is either a UCSC knownGene transcript ID (genic annotations) or a Ensembl transcript ID (lncRNA annotations), the `gene_id` is the Entrez ID, the `symbol` is the gene symbol from the `org.*.eg.db` mapping from the Entrez ID, and the `type` is of the form `[genome]_[type]_[name]`.

Examples

```
# Example with hg19 gene promoters
annots = c('hg19_genes_promoters')
annots_gr = build_annotations(genome = 'hg19', annotations = annots)

# See vignette for an example with custom annotation
```

---

build_cpg_annots *A helper function to build CpG related annotations.*

Description

Using the `AnnotationHub` package, extract CpG island track for the appropriate `genome` and construct the shores, shelves, and interCGI annotations as desired.

Usage

```
build_cpg_annots(
  genome = annotatr::builtin_genomes(),
  annotations = annotatr::builtin_annotations()
)
```

Arguments

genome	The genome assembly.
annotations	A character vector with entries of the form [genome]_cpg_{islands,shores,shelves,inter}.

Value

A list of GRanges objects.

---

build_enhancer_annots *A helper function to build enhancer annotations for hg19 and mm10 from FANTOM5.*

Description

A helper function to build enhancer annotations for hg19 and mm10 from FANTOM5.

Usage

```
build_enhancer_annots(genome = c("hg19", "hg38", "mm9", "mm10"))
```

Arguments

genome	The genome assembly.

Value

A GRanges object.

---

build_gene_annots *A helper function to build genic annotations.*

Description

Using the TxDb.* group of packages, construct genic annotations consisting of any combination of 1-5kb upstream of a TSS, promoters (< 1kb from TSS), 5UTRs, CDS, exons, first exons, introns, intron/exon and exon/intron boundaries, 3UTRs, and intergenic.

Usage

```
build_gene_annots(
  genome = annotatr::builtin_genomes(),
  annotations = annotatr::builtin_annotations()
)
```

Arguments

genome	The genome assembly.
annotations	A character vector with entries of the form [genome]_genes_{1to5kb,promoters,5UTRs,cds,exons,f

Value

A list of GRanges objects with unique id of the form [type]:i, tx_id being the UCSC knownGene transcript name, gene_id being the Entrez Gene ID, symbol being the gene symbol from the Entrez ID to symbol mapping in org.db for that species, and type being the annotation type.

---

build_hmm_annots — *A helper function to build chromHMM annotations for hg19 from UCSC Genome Browser.*

Description

A helper function to build chromHMM annotations for hg19 from UCSC Genome Browser.

Usage

```
build_hmm_annots(
  genome = c("hg19"),
  annotations = annotatr::builtin_annotations()
)
```

Arguments

genome	The genome assembly.
annotations	A character vector of valid chromatin state annotatin codes.

Value

A GRanges object.

---

build_lncrna_annots — *A helper function to build lncRNA annotations.*

Description

Using the AnnotationHub package, retrieve transcript level lncRNA annotations for either human (GRCh38) or mouse (GRCm38). If the genome is 'hg19', use the permalink from GENCODE and rtracklayer::import() to download and process.

Usage

```
build_lncrna_annots(genome = c("hg19", "hg38", "mm10"))
```

Arguments

genome The genome assembly.

Value

A GRanges object with id giving the transcript_type from the GENCODE file, tx_id being the Ensembl transcript ID, gene_id being the Entrez ID coming from a mapping of gene symbol to Entrez ID, symbol being the gene_name from the GENCODE file, and the type being [genome]_lncrna_gencode.

---

builtin_annotations *Function listing which annotations are available.*

Description

This includes the shortcuts. The expand_annotations() function helps handle the shortcuts.

Usage

```
builtin_annotations()
```

Value

A character vector of available annotations.

Examples

```
builtin_annotations()
```

---

builtin_genomes *Function returning supported TxDb.* genomes*

Description

Function returning supported TxDb.* genomes

Usage

```
builtin_genomes()
```

Value

A character vector of genomes for supported TxDb.* packages

Examples

```
builtin_genomes()
```

check_annotations *Function to check for valid annotations*

Description

Gives errors if any annotations are not in builtin_annotations() (and they are not in the required custom format), basicgenes are used, or the genome prefixes are not the same for all annotations.

Usage

```
check_annotations(annotations)
```

Arguments

annotations	A character vector of annotations possibly using the shortcuts

Value

If all the checks on the annotations pass, returns NULL to allow code to move forward.

expand_annotations *Function to expand annotation shortcuts*

Description

Function to expand annotation shortcuts

Usage

```
expand_annotations(annotations)
```

Arguments

annotations	A character vector of annotations, possibly using the shortcut accessors

Value

A vector of data accession-ized names that are ordered from upstream to downstream in the case of knownGenes and islands to interCGI in the case of cpgs.

get_cellline_from_code

Function to return cell line from chromatin annotation code

Description

Function to return cell line from chromatin annotation code

Usage

```
get_cellline_from_code(code)
```

Arguments

code	The annotation code, used in `build_annotations()`.

Value

A string of the cell line used in a chromatin annotation code

get_cellline_from_shortcut

Function to return cell line from chromatin annotation shortcut

Description

Function to return cell line from chromatin annotation shortcut

Usage

```
get_cellline_from_shortcut(shortcut)
```

Arguments

shortcut	The annotation shortcut, used in `build_annotations()`.

Value

A string of the cell line used in a chromatin annotation shortcut

get_orgdb_name — *Function to get correct org.* package name based on genome*

Description

Function to get correct org.* package name based on genome

Usage

```
get_orgdb_name(genome = annotatr::builtin_genomes())
```

Arguments

genome	A string giving the genome assembly.

Value

A string giving the correct org for org.db packages. e.g. hg19 -> Hs.

get_txdb_name — *Function to get correct TxDb.* package name based on genome*

Description

Function to get correct TxDb.* package name based on genome

Usage

```
get_txdb_name(genome = annotatr::builtin_genomes())
```

Arguments

genome	A string giving the genome assembly.

Value

A string giving the name of the correct TxDb.* package name based on genome.

`plot_annotation` *Plot the number of regions per annotation*

Description

Given a `GRanges` of annotated regions, plot the number of regions with the corresponding genomic annotations used in `annotation_order`. If a region is annotated to multiple annotations of the same `annot.type`, the region will only be counted once in the corresponding bar plot. For example, if a region were annotated to multiple exons, it would only count once toward the exon bar in the plot, but if it were annotated to an exon and an intron, it would count towards both.

Usage

```
plot_annotation(
  annotated_regions,
  annotated_random,
  annotation_order = NULL,
  plot_title,
  x_label,
  y_label,
  quiet = FALSE
)
```

Arguments

`annotated_regions`
: The `GRanges` result of `annotate_regions()`.

`annotated_random`
: The `GRanges` result of `annotate_regions()` on the randomized regions created from `randomize_regions()`.

`annotation_order`
: A character vector which doubles as the subset of annotations desired for the plot as well as the ordering. If `NULL`, all annotations are displayed.

`plot_title`
: A string used for the title of the plot. If missing, no title is displayed.

`x_label`
: A string used for the x-axis label. If missing, no x-axis label is displayed.

`y_label`
: A string used for the y-axis label. If missing, no y-axis label is displayed.

`quiet`
: Print progress messages (FALSE) or not (TRUE).

Value

A `ggplot` object which can be viewed by calling it, saved with `ggplot2::ggsave`, or edited.

Examples

```
######################################################################
# An example of ChIP-seq peaks with signalValue used for score

# Get premade CpG annotations
data('annotations', package = 'annotatr')

chip_bed = system.file('extdata', 'Gm12878_Stat3_chr2.bed.gz', package = 'annotatr')
chip_regions = read_regions(con = chip_bed, genome = 'hg19')

chip_rnd = randomize_regions(regions = chip_regions)

chip_annots = annotate_regions(
    regions = chip_regions,
    annotations = annotations,
    ignore.strand = TRUE)

chip_rnd_annots = annotate_regions(
    regions = chip_rnd,
    annotations = annotations,
    ignore.strand = TRUE)

annots_order = c(
    'hg19_cpg_islands',
    'hg19_cpg_shores')

p_annots = plot_annotation(annotated_regions = chip_annots,
    annotation_order = annots_order)
p_annots_rnd = plot_annotation(annotated_regions = chip_annots,
    annotated_random = chip_rnd_annots, annotation_order = annots_order)
```

`plot_categorical` *Plot a categorical data variable over another*

Description

Given a `GRanges` of annotated regions from `annotate_regions()`, visualize the the distribution of categorical data `fill` in categorical data `x`. A bar representing the distribution of all `fill` in `x` will be added according to the contents of `fill`. This is the distribution over all values of `x`. Additionally, when `annotated_random` is not missing, a "Random Regions" bar shows the distribution of random regions over `fill`.

Usage

```
plot_categorical(
  annotated_regions,
  annotated_random,
  x,
```

```
  fill = NULL,
  x_order = NULL,
  fill_order = NULL,
  position = "stack",
  plot_title,
  legend_title,
  x_label,
  y_label,
  quiet = FALSE
)
```

Arguments

annotated_regions
: The GRanges result of annotate_regions().

annotated_random
: The GRanges result of annotate_regions() on the randomized regions created from randomize_regions(). Random regions can only be used with fill == 'annot.type'.

x
: One of 'annot.type' or a categorical data column, indicating whether annotation classes or data classes will appear on the x-axis.

fill
: One of 'annot.type', a categorical data column, or NULL, indicating whether annotation classes or data classes will fill the bars. If NULL then the bars will be the total counts of the x classes.

x_order
: A character vector that subsets and orders the x classes. Default NULL, uses existing values.

fill_order
: A character vector that subsets and orders the fill classes. Default NULL, uses existing values.

position
: A string which has the same possible values as in ggplot2::geom_bar(..., position), i.e., 'stack', 'fill', 'dodge', etc.

plot_title
: A string used for the title of the plot. If missing, no title is displayed.

legend_title
: A string used for the legend title to describe fills (if fill is not NULL). Default displays corresponding variable name.

x_label
: A string used for the x-axis label. If missing, corresponding variable name used.

y_label
: A string used for the y-axis label. If missing, corresponding variable name used.

quiet
: Print progress messages (FALSE) or not (TRUE).

Details

For example, if a differentially methylated region has the categorical label hyper, and is annotated to a promoter, a 5UTR, two exons, and an intron. Each annotation will appear in the All bar once. Likewise for the hyper bar if the differential methylation status is chosen as x with annot.type chosen as fill.

Value

A ggplot object which can be viewed by calling it, or saved with ggplot2::ggsave.

Examples

```
# Get premade CpG annotations
data('annotations', package = 'annotatr')

dm_file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
extraCols = c(diff_meth = 'numeric', mu1 = 'numeric', mu0 = 'numeric')
dm_regions = read_regions(con = dm_file, extraCols = extraCols, genome = 'hg19',
    rename_score = 'pval', rename_name = 'DM_status', format = 'bed')
dm_regions = dm_regions[1:1000]

dm_annots = annotate_regions(
    regions = dm_regions,
    annotations = annotations,
    ignore.strand = TRUE)

dm_order = c(
    'hyper',
    'hypo')
cpg_order = c(
    'hg19_cpg_islands',
    'hg19_cpg_shores',
    'hg19_cpg_shelves',
    'hg19_cpg_inter')

dm_vn = plot_categorical(
    annotated_regions = dm_annots,
    x = 'DM_status',
    fill = 'annot.type',
    x_order = dm_order,
    fill_order = cpg_order,
    position = 'fill',
    legend_title = 'knownGene Annotations',
    x_label = 'DM status',
    y_label = 'Proportion')

# Create randomized regions
dm_rnd_regions = randomize_regions(regions = dm_regions)
dm_rnd_annots = annotate_regions(
    regions = dm_rnd_regions,
    annotations = annotations,
    ignore.strand = TRUE)

dm_vn_rnd = plot_categorical(
    annotated_regions = dm_annots,
    annotated_random = dm_rnd_annots,
    x = 'DM_status',
    fill = 'annot.type',
    x_order = dm_order,
    fill_order = cpg_order,
    position = 'fill',
    legend_title = 'knownGene Annotations',
    x_label = 'DM status',
```

```
y_label = 'Proportion')
```

`plot_coannotations` *Plot pair-wise annotations across regions*

Description

All co-occurring annotations associated with a region are computed and displayed as a heatmap.

Usage

```
plot_coannotations(
  annotated_regions,
  annotation_order = NULL,
  plot_title,
  axes_label,
  quiet = FALSE
)
```

Arguments

`annotated_regions`
: The `GRanges` result of `annotate_regions()`.

`annotation_order`
: A character vector which doubles as the subset of annotations desired for plot as well as the ordering. If `NULL`, all annotations are displayed.

`plot_title`
: A string used for the title of the plot. If missing, no plot title label is displayed.

`axes_label`
: A string used for the axis labels. If missing, corresponding variable name used.

`quiet`
: Print progress messages (FALSE) or not (TRUE).

Details

As with `plot_annotation()`, the number in each cell is the number of unique regions annotated to the pair of annotations.

For example, if a region is annotated to both a CpG shore and to two different exons simultaneously, the region will only be counted once in the CpG shore / exon cell. NOTE, this same region will count once in both the CpG shore and exon cells on the diagonal.

Value

A `ggplot` object which can be viewed by calling it, saved with `ggplot2::ggsave`, or edited.

Examples

```
# Get premade CpG annotations
data('annotations', package = 'annotatr')

dm_file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
extraCols = c(diff_meth = 'numeric', mu1 = 'numeric', mu0 = 'numeric')
dm_regions = read_regions(con = dm_file, extraCols = extraCols,
    rename_score = 'pval', rename_name = 'DM_status', format = 'bed')
dm_regions = dm_regions[1:1000]

dm_annots = annotate_regions(
    regions = dm_regions,
    annotations = annotations,
    ignore.strand = TRUE)

all_order = c(
    'hg19_cpg_islands',
    'hg19_cpg_shores',
    'hg19_cpg_shelves',
    'hg19_cpg_inter')

dm_vs_ca = plot_coannotations(
    annotated_regions = dm_annots,
    annotation_order = all_order,
    axes_label = 'Annotations',
    plot_title = 'Co-occurrence of Annotations')
```

`plot_numerical` *Plot numerical data over regions or regions summarized over annotations*

Description

This function produces either histograms over `facet`, or x-y scatterplots over `facet`. In the case of histograms over facets, the All distribution (hollow histogram with red outline) is the distribution of `x` over all the regions in the data. The facet specific distributions (solid gray) are the distribution of `x` over the regions in each facet. For example, a CpG with associated percent methylation annotated to a CpG island and a promoter will count once in the All distribution, but will count once each in the CpG island and promoter facet distributions.

Usage

```
plot_numerical(
  annotated_regions,
  x,
  y,
  facet,
  facet_order,
```

```
  bin_width = 10,
  plot_title,
  x_label,
  y_label,
  legend_facet_label,
  legend_cum_label,
  quiet = FALSE
)
```

Arguments

annotated_regions
: A GRanges returned from annotate_regions(). If the data is not summarized, the data is at the region level. If it is summarized, it represents the average or standard deviation of the regions by the character vector used for by in summarize_numerical().

x
: A string indicating the column of the GRanges to use for the x-axis.

y
: A string indicating the column of the GRanges to use for the y-axis. If missing, a a histogram over x will be plotted. If not missing, a scatterplot is plotted.

facet
: A string, or character vector of two strings, indicating indicating which categorical variable(s) in the GRanges to make ggplot2 facets over. When two facets are given, the first entry is the vertical facet and the second entry is the horizontal facet. Default is annot.type.

facet_order
: A character vector, or list of character vectors if facet has length 2, which gives the order of the facets, and can be used to subset the column in the GRanges used for the facet. For example, if facet = 'annot.type', then the annotations maybe subsetted to just CpG annotations. Default is NULL, meaning all annotations in their default order are used.

bin_width
: An integer indicating the bin width of the histogram used for score. Default 10. Select something appropriate for the data. NOTE: This is only used if y is NULL.

plot_title
: A string used for the title of the plot. If missing, no title is displayed.

x_label
: A string used for the x-axis label. If missing, no x-axis label is displayed.

y_label
: A string used for the y-axis label. If missing, no y-axis label is displayed.

legend_facet_label
: A string used to label the gray bar portion of the legend. Defaults to "x in facet".

legend_cum_label
: A string used to label the red outline portion of the legend. Defaults to "All in x".

quiet
: Print progress messages (FALSE) or not (TRUE).

Value

A ggplot object which can be viewed by calling it, or saved with ggplot2::ggsave.

Examples

```
# An example with multi-columned data

# Get premade CpG annotations
data('annotations', package = 'annotatr')

dm_file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
extraCols = c(diff_meth = 'numeric', mu1 = 'numeric', mu0 = 'numeric')
dm_regions = read_regions(con = dm_file, extraCols = extraCols,
    rename_score = 'pval', rename_name = 'DM_status', format = 'bed')
dm_regions = dm_regions[1:1000]

# Annotate the regions
dm_annots = annotate_regions(
    regions = dm_regions,
    annotations = annotations,
    ignore.strand = TRUE)

# Plot histogram of group 1 methylation rates across the CpG annotations.
# NOTE: Overall distribution (everything in \code{facet_order})
# is plotted in each facet for comparison.
dm_vs_regions_mu1 = plot_numerical(
    annotated_regions = dm_annots,
    x = 'mu1',
    facet = 'annot.type',
    facet_order = c('hg19_cpg_islands','hg19_cpg_shores',
        'hg19_cpg_shelves','hg19_cpg_inter'),
    bin_width = 5,
    plot_title = 'Group 1 Methylation over CpG Annotations',
    x_label = 'Group 1 Methylation')

# Plot histogram of group 1 methylation rates across the CpG annotations
# crossed with DM_status
dm_vs_regions_diffmeth = plot_numerical(
    annotated_regions = dm_annots,
    x = 'diff_meth',
    facet = c('annot.type','DM_status'),
    facet_order = list(
        c('hg19_genes_promoters','hg19_genes_5UTRs','hg19_cpg_islands'),
        c('hyper','hypo','none')),
    bin_width = 5,
    plot_title = 'Group 0 Region Methylation In Genes',
    x_label = 'Methylation Difference')

# Can also use the result of annotate_regions() to plot two numerical
# data columns against each other for each region, and facet by annotations.
dm_vs_regions_annot = plot_numerical(
    annotated_regions = dm_annots,
    x = 'mu0',
    y = 'mu1',
    facet = 'annot.type',
    facet_order = c('hg19_cpg_islands','hg19_cpg_shores',
```

```
            'hg19_cpg_shelves','hg19_cpg_inter'),
        plot_title = 'Region Methylation: Group 0 vs Group 1',
        x_label = 'Group 0',
        y_label = 'Group 1')

    # Another example, but using differential methylation status as the facets.
    dm_vs_regions_name = plot_numerical(
        annotated_regions = dm_annots,
        x = 'mu0',
        y = 'mu1',
        facet = 'DM_status',
        facet_order = c('hyper','hypo','none'),
        plot_title = 'Region Methylation: Group 0 vs Group 1',
        x_label = 'Group 0',
        y_label = 'Group 1')
```

`plot_numerical_coannotations`

Plot numerical data occurring in pairs of annotations

Description

Plot numerical data associated with regions occurring in `annot1`, `annot2` and in both. As with `plot_numerical()`, the result is a plot of histograms or x-y scatterplots.

Usage

```
plot_numerical_coannotations(
  annotated_regions,
  x,
  y,
  annot1,
  annot2,
  bin_width = 10,
  plot_title,
  x_label,
  y_label,
  legend_facet_label,
  legend_cum_label,
  quiet = FALSE
)
```

Arguments

`annotated_regions`
: A `GRanges` returned from `annotate_regions()`.

`x`
: A string indicating the column of the `GRanges` to use for the x-axis.

y	A string indicating the column of the GRanges to use for the y-axis. If missing, a histogram over x will be plotted. If not missing, a scatterplot is plotted.
annot1	A string indicating the first annotation type.
annot2	A string indicating the second annotation type.
bin_width	An integer indicating the bin width of the histogram used for score. Default 10. Select something appropriate for the data. NOTE: This is only used if y is NULL.
plot_title	A string used for the title of the plot. If missing, no title is displayed.
x_label	A string used for the x-axis label. If missing, no x-axis label is displayed.
y_label	A string used for the y-axis label. If missing, no y-axis label is displayed.
legend_facet_label	A string used to label the gray bar portion of the legend. Defaults to "x in annot pair".
legend_cum_label	A string used to label the red outline portion of the legend. Defaults to "All x".
quiet	Print progress messages (FALSE) or not (TRUE).

Details

For example, a CpG with associated percent methylation annotated to a CpG island and a promoter will count once in the All distribution and once in the CpG island / promoter facet distribution. However, a CpG associated only with a promoter will count once in the All distribution and once in the promoter / promoter distribution.

Value

A ggplot object which can be viewed by calling it, or saved with ggplot2::ggsave.

Examples

```
# Get premade CpG annotations
data('annotations', package = 'annotatr')

dm_file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
extraCols = c(diff_meth = 'numeric', mu1 = 'numeric', mu0 = 'numeric')
dm_regions = read_regions(con = dm_file, extraCols = extraCols,
    rename_score = 'pval', rename_name = 'DM_status', format = 'bed')
dm_regions = dm_regions[1:1000]

dm_annots = annotate_regions(
    regions = dm_regions,
    annotations = annotations,
    ignore.strand = TRUE)

dm_vs_num_co = plot_numerical_coannotations(
    annotated_regions = dm_annots,
    x = 'mu0',
    annot1 = 'hg19_cpg_islands',
    annot2 = 'hg19_cpg_shelves',
```

```
bin_width = 5,
plot_title = 'Group 0 Perc. Meth. in CpG Islands and Promoters',
x_label = 'Percent Methylation')
```

---

randomize_regions *Randomize Regions*

Description

randomize_regions is a wrapper function for regioneR::randomizeRegions() that simplifies the creation of randomized regions for an input set of regions read with read_regions(). It relies on the seqlengths of regions in order to build the appropriate genome object for regioneR::randomizeRegions().

Usage

```
randomize_regions(
  regions,
  allow.overlaps = TRUE,
  per.chromosome = TRUE,
  quiet = FALSE
)
```

Arguments

regions	A GRanges object from read_regions.
allow.overlaps	A logical stating whether random regions can overlap input regions (TRUE) or not (FALSE). Default TRUE.
per.chromosome	A logical stating whether the random regions should remain on the same chromosome (TRUE) or not (FALSE). Default TRUE.
quiet	Print progress messages (FALSE) or not (TRUE).

Details

NOTE: The data associated with the input regions are not passed on to the random regions.

Value

A GRanges object of randomized regions based on regions from read_regions(). NOTE: Data associated with the original regions is not attached to the randomized regions.

Examples

```
# Create random region set based on ENCODE ChIP-seq data
file = system.file('extdata', 'Gm12878_Stat3_chr2.bed.gz', package = 'annotatr')
r = read_regions(con = file, genome = 'hg19')

random_r = randomize_regions(regions = r)
```

read_annotations *Read custom annotations*

Description

`read_annotations()` is a wrapper for the `rtracklayer::import()` function that creates a `GRanges` object matching the structure of annotations built with `build_annotations()`. The structure is defined by `GRanges`, with the `mcols()` with names `c('id','gene_id','symbol','type')`.

Usage

```
read_annotations(con, name, genome = NA, format, extraCols = character(), ...)
```

Arguments

`con`	A path, URL, connection or BEDFile object. See `rtracklayer::import.bed()` documentation.
`name`	A string for the name of the annotations to be used in the name of the object, [genome]_custom_[name]
`genome`	From `rtracklayer::import()`: The identifier of a genome, or NA if unknown. Typically, this is a UCSC identifier like 'hg19'. An attempt will be made to derive the `seqinfo` on the return value using either an installed BSgenome package or UCSC, if network access is available.
`format`	From `rtracklayer::import()`: The format of the output. If not missing, should be one of 'bed', 'bed15', 'bedGraph' or 'bedpe'. If missing and 'con' is a filename, the format is derived from the file extension. This argument is unnecessary when 'con' is a derivative of 'RTLFile'.
`extraCols`	From `rtracklayer::import.bed()`: A character vector in the same form as 'colClasses' from 'read.table'. It should indicate the name and class of each extra/special column to read from the BED file. As BED does not encode column names, these are assumed to be the last columns in the file. This enables parsing of the various BEDX+Y formats.
`...`	Parameters to pass onto the format-specific method of `rtracklayer::import()`.

Value

A `GRanges` object stored in `annotatr_cache`. To view a custom annotation, do `annotatr_cache$get(name)`. To add a custom annotation to the set of annotations, include `'[genome]_custom_[name]'` in the call to `build_annotations()`. See example below.

Examples

```
# Read in a BED3 file as a custom annotation
file = system.file('extdata', 'test_annotations_3.bed', package='annotatr')
read_annotations(con = file, name = 'test', genome = 'hg19')
build_annotations(genome = 'hg19', annotations = 'hg19_custom_test')
```

```
print(annotatr_cache$get('hg19_custom_test'))
```

`read_regions` *Read genomic regions in BEDX+Y format*

Description

`read_regions()` reads genomic regions by calling the `rtracklayer::import()` function. This function can automatically deal with BEDX files from BED3 to BED6. For BED6+Y, the `extraCols` argument should be used to correctly interpret the extra columns.

Usage

```
read_regions(
  con,
  genome = NA,
  format,
  extraCols = character(),
  rename_name,
  rename_score,
  ...
)
```

Arguments

`con`	A path, URL, connection or BEDFile object. See `rtracklayer::import()` documentation.
`genome`	From `rtracklayer::import()`: The identifier of a genome, or NA if unknown. Typically, this is a UCSC identifier like 'hg19'. An attempt will be made to derive the `seqinfo` on the return value using either an installed BSgenome package or UCSC, if network access is available.
`format`	From `rtracklayer::import()`: The format of the output. If not missing, should be one of 'bed', 'bed15', 'bedGraph' or 'bedpe'. If missing and 'con' is a filename, the format is derived from the file extension. This argument is unnecessary when 'con' is a derivative of 'RTLFile'.
`extraCols`	From `rtracklayer::import()`: A character vector in the same form as 'colClasses' from 'read.table'. It should indicate the name and class of each extra/special column to read from the BED file. As BED does not encode column names, these are assumed to be the last columns in the file. This enables parsing of the various BEDX+Y formats.
`rename_name`	A string to rename the name column of the BED file. For example, if the name column actually contains a categorical variable.
`rename_score`	A string to rename the score column of the BED file. For example, if the score column represents a quantity about the data besides the score in the BED specification.
`...`	Parameters to pass onto the format-specific method of `rtracklayer::import()`.

Details

NOTE: The name (4th) and score (5th) columns are so named. If these columns have a particular meaning for your data, they should be renamed with the rename_name and/or rename_score parameters.

Value

A GRanges object.

Examples

```
 # Example of reading a BED6+3 file where the last 3 columns are non-standard
file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
 extraCols = c(diff_meth = 'numeric', mu0 = 'numeric', mu1 = 'numeric')
 gr = read_regions(con = file, genome = 'hg19', extraCols = extraCols, format = 'bed',
     rename_name = 'DM_status', rename_score = 'pval')
```

---

reformat_hmm_codes *Function to recode classes from chromHMM type column*

Description

Function to recode classes from chromHMM type column

Usage

```
reformat_hmm_codes(hmm_codes)
```

Arguments

hmm_codes	in the original form from UCSC Genome Browser track.

Value

A character vector of chromHMM classes with numbers and underscores removed.

subset_order_tbl *Function to subset a tbl_df or grouped_df by a column*

Description

Function to subset a tbl_df or grouped_df by a column

Usage

```
subset_order_tbl(tbl, col, col_order)
```

Arguments

tbl	A tbl_df or grouped_df.
col	A string indicating which column of of tbl to subset and order
col_order	A character vector indicating the order of col.

Value

A modified version of summary with col subsetted by col_order.

summarize_annotations *Summarize annotation counts*

Description

Given a GRanges of annotated regions, count the number of regions in each annotation type. If annotated_random is not NULL, then the same is computed for the random regions.

Usage

```
summarize_annotations(annotated_regions, annotated_random, quiet = FALSE)
```

Arguments

annotated_regions	The GRanges result of annotate_regions().
annotated_random	The GRanges result of annotate_regions() on the randomized regions created from randomize_regions().
quiet	Print progress messages (FALSE) or not (TRUE).

Details

If a region is annotated to multiple annotations of the same `annot.type`, the region will only be counted once. For example, if a region were annotated to multiple exons, it would only count once toward the exons, but if it were annotated to an exon and an intron, it would count towards both.

Value

A `tbl_df` of the number of regions per annotation type.

Examples

```
### An example of ChIP-seq peaks with signalValue

# Get premade CpG annotations
data('annotations', package = 'annotatr')

file = system.file('extdata', 'Gm12878_Stat3_chr2.bed.gz', package = 'annotatr')
r = read_regions(con = file, genome = 'hg19')

a = annotate_regions(
    regions = r,
    annotations = annotations,
    ignore.strand = TRUE,
    quiet = FALSE)

rnd = randomize_regions(regions = r)

rnd_annots = annotate_regions(
    regions = rnd,
    annotations = annotations,
    ignore.strand = TRUE,
    quiet = FALSE)

# Summarize the annotated regions without randomized regions
s = summarize_annotations(annotated_regions = a)

# Summarize the annotated regions with randomized regions
s_rnd = summarize_annotations(
    annotated_regions = a,
    annotated_random = rnd_annots)
```

`summarize_categorical` *Summarize categorical data over groupings of annotated regions*

Description

Given a `GRanges` of annotated regions, count the number of regions when the annotations are grouped by categorical columns.

Usage

```
summarize_categorical(
  annotated_regions,
  by = c("annot.type", "annot.id"),
  quiet = FALSE
)
```

Arguments

annotated_regions
: The GRanges result of annotate_regions().

by
: A character vector to group the data in as.data.frame(annotated_regions) by and tally over. Default is c('annot.type', 'annot.id').

quiet
: Print progress messages (FALSE) or not (TRUE).

Details

If a region is annotated to multiple annotations of the same annot.type, the region will only be counted once. For example, if a region were annotated to multiple exons, it would only count once toward the exons, but if it were annotated to an exon and an intron, it would count towards both.

Value

A grouped dplyr::tbl_df of the counts of groupings according to the by vector.

Examples

```
 # Get premade CpG annotations
 data('annotations', package = 'annotatr')

r_file = system.file('extdata', 'test_read_multiple_data_nohead.bed', package='annotatr')
extraCols = c(pval = 'numeric', mu1 = 'integer', mu0 = 'integer', diff_exp = 'character')
r = read_regions(con = r_file, genome = 'hg19', extraCols = extraCols, rename_score = 'coverage')

 a = annotate_regions(
     regions = r,
     annotations = annotations,
     ignore.strand = TRUE)

 sc = summarize_categorical(
     annotated_regions = a,
     by = c('annot.type', 'name'),
     quiet = FALSE)
```

summarize_numerical *Summarize numerical data over groupings of annotated regions*

Description

Given a GRanges of annotated regions, summarize numerical data columns based on a grouping.

Usage

```
summarize_numerical(
  annotated_regions,
  by = c("annot.type", "annot.id"),
  over,
  quiet = FALSE
)
```

Arguments

annotated_regions
: The GRanges result of annotate_regions().

by
: A character vector of the columns of as.data.frame(annotated_regions) to group over. Default is c(annot.type, annot.id).

over
: A character vector of the numerical columns in as.data.frame(annotated_regions) to count, take the mean, and take the sd over after grouping according to the by column. NOTE: If more than one value is used, the naming scheme for the resuling dplyr::tbl summary columns are COLNAME_n, COLNAME_mean, COLNAME_sd. If over has length one, then the column names are n, mean, sd.

quiet
: Print progress messages (FALSE) or not (TRUE).

Details

NOTE: We do not take the distinct values of seqnames, start, end, annot.type as in the other summarize_*() functions because in the case of a region that intersected two distinct exons, using distinct() would destroy the information of the mean of the numerical column over one of the exons, which is not desirable.

Value

A grouped dplyr::tbl_df, and the count, mean, and sd of the cols by the groupings.

Examples

```
### Test on a very simple bed file to demonstrate different options

# Get premade CpG annotations
data('annotations', package = 'annotatr')
```

```
r_file = system.file('extdata', 'test_read_multiple_data_nohead.bed', package='annotatr')
extraCols = c(pval = 'numeric', mu1 = 'integer', mu0 = 'integer', diff_exp = 'character')
r = read_regions(con = r_file, genome = 'hg19', extraCols = extraCols, rename_score = 'coverage')

a = annotate_regions(
       regions = r,
       annotations = annotations,
       ignore.strand = TRUE)

# Testing over normal by
sn1 = summarize_numerical(
       annotated_regions = a,
       by = c('annot.type', 'annot.id'),
       over = c('coverage', 'mu1', 'mu0'),
       quiet = FALSE)

# Testing over a different by
sn2 = summarize_numerical(
       annotated_regions = a,
       by = c('diff_exp'),
       over = c('coverage', 'mu1', 'mu0'))
```

`tidy_annotations` *Function to tidy up annotation accessors for visualization*

Description

Function to tidy up annotation accessors for visualization

Usage

```
tidy_annotations(annotations)
```

Arguments

`annotations`	A character vector of annotations, in the order they are to appear in the visualization.

Value

A list of mappings from original annotation names to names ready for visualization.

Index