Package: annotatr (via r-universe)

November 1, 2026

Title Annotation of Genomic Regions to Genomic Annotations

Version 1.39.19

Date 2025-06-21

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 (>= 4.5.0)

Imports AnnotationDbi, AnnotationHub, BiocFileCache, dplyr, GenomicFeatures (>= 1.61.4), GenomicRanges (>= 1.61.1), Seqinfo, ggplot2 (>= 3.5.0), IRanges, methods, readr, regioneR, reshape2, rlang, rtracklayer (>= 1.69.1), S4Vectors (>= 0.23.10), stats, utils

Suggests GenomeInfoDb, BiocStyle, curl, devtools, ensembldb, knitr, org.Dm.eg.db, org.Dr.eg.db, org.Gg.eg.db, org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db, rmarkdown, roxygen2, testthat (>= 3.2.0), txdbmaker, withr, TxDb.Dmelanogaster.UCSC.dm3.ensGene, TxDb.Dmelanogaster.UCSC.dm6.ensGene, TxDb.Drerio.UCSC.danRer10.refGene, TxDb.Drerio.UCSC.danRer11.refGene, 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.Mmusculus.UCSC.mm39.knownGene, TxDb.Rnorvegicus.UCSC.rn4.ensGene, TxDb.Rnorvegicus.UCSC.rn5.refGene, TxDb.Rnorvegicus.UCSC.rn6.refGene, TxDb.Rnorvegicus.UCSC.rn7.refGene

VignetteBuilder knitr

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

License GPL-3

LazyData true

RoxygenNote 7.3.3

Encoding UTF-8

Config/testthat/edition 3

biocViews Software, Annotation, GenomeAnnotation, FunctionalGenomics, Visualization

Config/pak/sysreqs make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-10-02 09:19:56 UTC

RemoteUrl https://github.com/bioc/annotatr

RemoteRef HEAD

RemoteSha 4567918a24fffbc2bbf78c0cb73fc82b444bca1c

Contents

annotate_regions . . . 3
annotations . . . 5
annotatr . . . 5
annotatr_cache . . . 6
build_ah_annots . . . 7
build_annotations . . . 7
build_ccre_annots . . . 9
build_cpg_annots . . . 10
build_enhancer_annots . . . 10
build_gene_annots . . . 11
build_hmm_annots . . . 12
build_lncrna_annots . . . 12
build_mane_txdb . . . 13
build_txdb_annotations . . . 13
build_txdb_gene_annots . . . 15
builtin_annotations . . . 15
builtin_annotations_table . . . 16
builtin_genomes . . . 17
cached-annotations . . . 17
check_annotations . . . 18
check_regions_genome . . . 19
clear_cached_annotations . . . 19
coannotation_pairs . . . 20
download_annotation_file . . . 21

expand_annotations . . . 21
get_annotation_rname . . . 22
get_bfc . . . 22
get_cache_dir . . . 23
get_cache_rids . . . 23
get_cellline_from_code . . . 24
get_cellline_from_shortcut . . . 24
get_chrom_aliases . . . 25
get_genark_seqinfo . . . 25
get_genark_url . . . 26
get_gene_table . . . 26
get_mane_summary . . . 27
get_orgdb_name . . . 27
get_txdb_name . . . 28
list_cached_annotations . . . 28
load_cached_annotation . . . 29
plot_annotation . . . 29
plot_categorical . . . 31
plot_coannotations . . . 33
plot_numerical . . . 35
plot_numerical_coannotations . . . 37
randomize_regions . . . 39
read_annotations . . . 40
read_ccre_bed . . . 41
read_regions . . . 42
reformat_hmm_codes . . . 43
save_cached_annotation . . . 44
set_genome_seqinfo . . . 44
standardize_mcols . . . 45
strip_id_version . . . 45
subset_order_tbl . . . 46
summarize_annotations . . . 46
summarize_categorical . . . 47
summarize_genes . . . 49
summarize_numerical . . . 50
tidy_annotations . . . 52
ucsc_seqlevels . . . 52

Index **53**

`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.

Author(s)

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

Other contributors:

- Maureen A. Sartor <sartorma@med.umich.edu> [thesis advisor]

See Also

Useful links:

- Report bugs at https://www.github.com/rcavalcante/annotatr/issues

---

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. `build_annotations()` gets custom annotations from it by name, e.g. `'hg19_custom_test'`. It lasts for the current R session only.

Usage

```
annotatr_cache
```

Format

An object of class `list` of length 3.

Details

It is usually simpler to use the `GRanges` that `read_annotations()` returns, and combine it with other annotations using `c()`.

`annotatr_cache` is separate from the cache on disk of annotations built by `build_annotations()`. See `cached-annotations`.

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, cache = TRUE)
```

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 "basicmane" shortcut (hg38 only) builds the same regions from MANE Select transcripts, and the "basiccanonical" shortcut from Ensembl canonical transcripts. The "cpgs" shortcut builds the following regions: CpG islands, shores, shelves, and interCGI regions. The "ccres" shortcut (hg38 and mm10) builds all 8 classes of ENCODE cCREs. NOTE: Shortcuts need to be appended by the genome, e.g. hg19_basicgenes.
cache	A logical stating whether to load annotations from, and save them to, the cache on disk (TRUE), or to build them from scratch without the cache (FALSE). See cached-annotations. 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.

Details

The [genome]_ccre_* annotations (hg38 and mm10) are ENCODE candidate cis-regulatory elements (cCREs) from the SCREEN registry (version 4), in 8 classes: PLS (promoter-like), pELS and dELS (proximal and distal enhancer-like), CA-H3K4me3, CA-CTCF, and CA-TF (chromatin accessible with H3K4me3, CTCF, or TF binding), CA (chromatin accessible only), and TF (TF binding only). The [genome]_ccres shortcut builds all 8. Their id is ENCODE's ID for the cCRE, its accession (e.g. EH38E2776516), which can be looked up at https://screen.wenglab.org. There are 2.3 million cCREs for hg38 (about 300 MB in memory) and 0.9 million for mm10, most of them dELS.

The [genome]_canonical_* annotations (hg38, mm39, rn7, danRer11, dm6, and oviariramb2) are the same gene annotations, built only from the Ensembl canonical transcript of each gene (all biotypes, e.g. protein-coding and lncRNA), from the Ensembl 113 EnsDb in AnnotationHub. For human, the canonical transcript is the MANE Select transcript when there is one, and otherwise is chosen by Ensembl from conservation, expression, APPRIS, UniProt, CDS length, and clinical variants. For other species, Ensembl chooses by biotype (protein-coding first) and then the longest combined exon length, so the canonical transcript is roughly the longest protein-coding transcript. Genes on alternate haplotypes and fix patches (_alt and _fix sequences), which Ensembl gives separate gene IDs, are left out. gene_id and ensembl_id are the Ensembl gene ID. The [genome]_basiccanonical shortcut builds the same types as basicgenes.

The hg38_mane_* annotations are the same gene annotations (except intergenic), built only from the MANE Select transcripts: one transcript per protein-coding gene, agreed on by NCBI and Ensembl (https://www.ncbi.nlm.nih.gov/refseq/MANE/). Use them to annotate regions to the main isoform of each gene, instead of every isoform in hg38_genes_*. MANE Plus Clinical transcripts are not included.

Value

A GRanges object of all the annotations combined. The mcols are id, tx_id, gene_id, symbol, entrez_id, ensembl_id, type:

id A unique name, e.g. promoter:1.

tx_id The transcript ID of gene annotations, from the source of the gene models (e.g. a UCSC knownGene or Ensembl transcript ID).

gene_id The gene ID used by the source of the gene models: the Entrez ID for most genomes, the Ensembl ID for rn4 and sheep (oviaramb2), and the FlyBase ID for dm3 and dm6.

symbol, entrez_id, ensembl_id The gene symbol, Entrez ID, and Ensembl ID (without version) of every gene annotation, whatever the source's gene ID is. They come from the org.*.eg.db package, the EnsDb, or the MANE summary. When a gene ID maps to more than one symbol, Entrez ID, or Ensembl ID, the first is used. They are NA for genes without a mapping, and for other annotations.

type The annotation code, of the form [genome]_[group]_[type].

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_ccre_annots *A helper function to build ENCODE cCRE annotations.*

Description

Candidate cis-regulatory elements (cCREs) from the ENCODE registry (SCREEN), version CCRE$version, in 8 classes: promoter-like (PLS), proximal and distal enhancer-like (pELS, dELS), chromatin accessible with H3K4me3, CTCF, or TF binding (CA-H3K4me3, CA-CTCF, CA-TF), chromatin accessible only (CA), and TF binding only (TF).

Usage

```
build_ccre_annots(
  genome = names(CCRE$files),
  annotations = annotatr::builtin_annotations(),
  cache = TRUE
)
```

Arguments

genome	The genome assembly, hg38 or mm10.
annotations	A character vector of annotations of the form [genome]_ccre_[class].
cache	A logical stating whether to use the cache on disk for downloads.

Value

A GRanges object whose id is the cCRE accession, e.g. EH38E2776516.

---

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(),
  cache = TRUE
)
```

Arguments

genome	The genome assembly.
annotations	A character vector with entries of the form [genome]_cpg_{islands,shores,shelves,inter}.
cache	A logical stating whether to use the cache on disk for downloads.

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"), cache = TRUE)
```

Arguments

genome	The genome assembly.
cache	A logical stating whether to use the cache on disk for downloads.

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. For genomes without a TxDb.* package (e.g. sheep, oviariramb2), an Ensembl EnsDb from AnnotationHub is used instead, and tx_id and gene_id are Ensembl IDs.

Usage

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

Arguments

genome	The genome assembly.
annotations	A character vector with entries of the form [genome]_genes_{1to5kb,promoters,5UTRs,cds,exons,f or hg38_mane_* with the same types except intergenic. All entries must be from the same group, genes or mane.
cache	A logical stating whether to use the cache on disk for downloads.

Details

The mane annotations (hg38 only) are built the same way from the MANE Select transcripts, one per protein-coding gene, with a TxDb made from the MANE GTF. Their tx_id is the Ensembl transcript ID, and gene_id is the Entrez ID, as for hg38_genes_*.

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(),
  cache = TRUE
)
```

Arguments

genome	The genome assembly.
annotations	A character vector of valid chromatin state annotatin codes.
cache	A logical stating whether to use the cache on disk for downloads.

Value

A GRanges object.

build_lncrna_annots *A helper function to build lncRNA annotations.*

Description

Download transcript level lncRNA annotations from GENCODE for human (hg19: release 19, hg38: release 31) or mouse (mm10: release M6), and process them with `rtracklayer::import()`.

Usage

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

Arguments

genome	The genome assembly.
cache	A logical stating whether to use the cache on disk for downloads.

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.

---

build_mane_txdb *Function to build a TxDb of the MANE Select transcripts*

Description

The transcripts come from the Ensembl version of the MANE GTF, and their gene IDs are replaced by Entrez IDs from the MANE summary.

Usage

```
build_mane_txdb(cache = TRUE)
```

Arguments

cache	A logical stating whether to use the cache on disk for downloads.

Value

A TxDb object.

---

build_txdb_annotations *Build gene annotations from any TxDb or EnsDb*

Description

Build the same gene annotations as build_annotations() (promoters, exons, introns, etc.) from gene models you provide, e.g. GENCODE or RefSeq gene models, or a genome without built-in annotations. The result can be combined with built-in annotations with c() and used with annotate_regions(), summarize_genes(), and the plotting functions.

Usage

```
build_txdb_annotations(
  txdb,
  genome,
  group,
  annotations = "basicgenes",
  orgdb = NULL,
  keytype = "ENTREZID"
)
```

Arguments

txdb	A TxDb or EnsDb object.
genome	A string giving the genome assembly of txdb, e.g. 'mm39'. If txdb records a genome, they must agree.
group	A string naming the source of the gene models, e.g. 'gencode' or 'refseq'. It must be letters and numbers only, and can't be a built-in group (genes, mane, cpg, enhancers, chromatin, lncrna, custom).
annotations	A character vector of annotation types to build: 1to5kb, promoters, 5UTRs, cds, exons, firstexons, introns, intronexonboundaries, exonintronboundaries, 3UTRs, and intergenic. The basicgenes shortcut (the default) builds 1-5kb upstream of TSSs, promoters, 5UTRs, exons, introns, and 3UTRs.
orgdb	An optional OrgDb object, e.g. org.Mm.eg.db::org.Mm.eg.db, to get gene symbols, Entrez IDs, and Ensembl IDs for the gene IDs of a TxDb. An EnsDb has them already.
keytype	A string giving the type of the gene IDs in txdb, as a keytype of orgdb, e.g. 'ENTREZID' (the default) or 'ENSEMBL'. For 'ENSEMBL', versions of gene IDs (e.g. the .12 in ENSMUSG00000000001.12) are ignored when looking up symbols.

Details

To build from a GTF or GFF3 file, first make a TxDb with txdbmaker::makeTxDbFromGFF(). An EnsDb can come from AnnotationHub or ensembldb::ensDbFromGtf().

The annotation codes are [genome]_[group]_[type], e.g. mm39_gencode_promoters, so annotations from different gene models can be used together. The chromosome names are those of the txdb. An EnsDb has Ensembl names (e.g. 1 rather than chr1), which can be changed with GenomeInfoDb::seqlevelsStyle(edb) = 'UCSC' before building, to match regions with UCSC names.

Annotations built with this function aren't cached on disk. See cached-annotations.

Value

A GRanges object with mcols id, tx_id, gene_id, symbol, entrez_id, ensembl_id, and type, as from build_annotations(). The tx_id and gene_id are the transcript names and gene IDs of txdb. The symbol, entrez_id, and ensembl_id come from an EnsDb, or from orgdb (the first, when a gene ID maps to more than one), and are NA without either.

See Also

build_annotations, read_annotations for annotations from BED files.

Examples

```
if(requireNamespace('TxDb.Hsapiens.UCSC.hg19.knownGene', quietly = TRUE) &&
    requireNamespace('org.Hs.eg.db', quietly = TRUE)) {
  # Any TxDb works, e.g. one made from a GTF with txdbmaker::makeTxDbFromGFF()
  txdb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
```

```
  annots = build_txdb_annotations(txdb, genome = 'hg19', group = 'knowngene',
      annotations = 'promoters', orgdb = org.Hs.eg.db::org.Hs.eg.db)

  unique(annots$type)
}
```

`build_txdb_gene_annots` *A helper function to build genic annotations from a TxDb or EnsDb*

Description

This does the work for `build_gene_annots()` and `build_txdb_annotations()`.

Usage

```
build_txdb_gene_annots(txdb, prefix, annotations, gene_table = NULL)
```

Arguments

`txdb`	A `TxDb` or `EnsDb` object.
`prefix`	A string giving the start of the annotation codes, `[genome]_[group]`, e.g. `'hg19_genes'`.
`annotations`	A character vector of annotation codes of the form `[prefix]_[type]`, where the types are `1to5kb`, `promoters`, `5UTRs`, `cds`, `exons`, `firstexons`, `introns`, `intronexonboundaries`, `exonintronboundaries`, `3UTRs`, and `intergenic`.
`gene_table`	A `data.frame` from `get_gene_table()`, with columns `gene_id`, `symbol`, `entrez_id`, and `ensembl_id`, for the gene IDs of `txdb`. If `NULL`, the gene names and Entrez IDs of an `EnsDb` are used, and those of a `TxDb` are `NA`.

Value

A `GRangesList` with one `GRanges` per annotation, with the standard `mcols` (see `standardize_mcols()`).

`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_annotations_table`

Table of builtin annotations by genome

Description

Summarizes `builtin_annotations()` as one row per genome and one column per group of annotations, showing which are available for which genome builds.

Usage

```
builtin_annotations_table()
```

Details

The columns and the shortcuts that build the whole group:

genes Gene annotations from all transcripts (`[genome]_basicgenes`, or e.g. `[genome]_genes_promoters`).

mane Gene annotations from MANE Select transcripts (`[genome]_basicmane`).

canonical Gene annotations from Ensembl canonical transcripts (`[genome]_basiccanonical`).

cpgs CpG islands, shores, shelves, and inter-CGI (`[genome]_cpgs`).

enhancers FANTOM5 enhancers (`[genome]_enhancers_fantom`).

chromatin chromHMM chromatin states for 9 cell lines (e.g. `[genome]_Gm12878-chromatin`).

lncrna GENCODE lncRNA transcripts (`[genome]_lncrna_gencode`).

ccres ENCODE candidate cis-regulatory elements (`[genome]_ccres`).

Value

A `data.frame` with a `genome` column, and a logical column for each group, in the order of `builtin_genomes()`.

Examples

```
builtin_annotations_table()
```

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()
```

cached-annotations *Cached annotations*

Description

`build_annotations()` saves each annotation it builds, and each file it downloads, in a cache on disk. Later calls load them from the cache instead of building them again, which is much faster (e.g. seconds instead of about half a minute for the hg19 gene annotations). This cache is separate from `annotatr_cache`, which holds custom annotations for the current R session only.

Details

The cache is in `tools::R_user_dir('annotatr', which = 'cache')` and is managed with the `BiocFileCache` package. To use another directory, set `options(annotatr.cache = '/path/to/cache')`, e.g. in your `.Rprofile`.

A cached annotation is rebuilt automatically after annotatr is updated, and for gene annotations, after the `TxDb.*` or `org.*.eg.db` package is updated. Downloaded files are not updated automatically, because their sources rarely change.

Value

None. This help page describes the cache.

Troubleshooting

See what is cached `list_cached_annotations()` lists each cached annotation and download, with its size and when it was added.

Annotations seem out of date, or a source has been updated Remove the cached annotations and downloads for the genome, e.g. `clear_cached_annotations(genome = 'hg19')`, and build again.

Build once without the cache `build_annotations(..., cache = FALSE)` builds from scratch, and neither reads nor writes the cache. This is useful to check whether a problem comes from the cache.

Errors reading a cached file A corrupted cached annotation (e.g. from an interrupted build) is removed and rebuilt automatically. If errors continue, remove the genome from the cache with `clear_cached_annotations(genome = ...)`.

A download fails Downloads are retried 3 times. A failed download isn't cached, so the next call tries again. Check the network connection, and that the source (e.g. UCSC, GENCODE, FANTOM5) is up.

The cache uses too much disk space Check sizes with `list_cached_annotations()`, and remove genomes you no longer need with `clear_cached_annotations(genome = ...)`. `clear_cached_annotations()` removes everything.

Home directory is small or read-only (e.g. on a cluster) Set `options(annotatr.cache = '/path/with/space')`.

"database is locked" errors The cache index is a SQLite database, which can lock when parallel jobs write to it at once. Build the annotations once before starting parallel jobs, so the jobs only read from the cache, or give each job its own cache directory with `options(annotatr.cache = ...)`.

Problems with AnnotationHub Some annotations come from AnnotationHub (e.g. hg19 CpG islands, sheep gene models), which has its own cache. See `AnnotationHub::hubCache()` for its location, and `AnnotationHub::removeResources()` or `AnnotationHub::removeCache()` to clear it.

See Also

`list_cached_annotations`, `clear_cached_annotations`, `build_annotations`

---

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.

---

check_regions_genome *Function to check regions and annotations are from the same genome*

Description

Gives an error if the regions and annotations have different genomes, or no chromosome names in common (e.g. 2 and chr2). If the regions have no genome, it can't check the genomes match, so it gives a message once per session suggesting read_regions(genome = ...).

Usage

```
check_regions_genome(regions, annotations, quiet = FALSE)
```

Arguments

regions	A GRanges object of regions.
annotations	A GRanges object of annotations.
quiet	Print the message about regions without a genome (FALSE) or not (TRUE).

Value

NULL, invisibly, if the checks pass.

---

clear_cached_annotations *Clear cached annotations and downloads*

Description

Remove annotations and downloaded files from the cache used by build_annotations(), so they will be built or downloaded again. See cached-annotations for more about the cache.

Usage

```
clear_cached_annotations(genome = NULL)
```

Arguments

genome	A character vector of genome assemblies to remove, e.g. 'hg19'. If NULL (the default), everything is removed.

Value

A data.frame of the removed items, as from list_cached_annotations(), invisibly.

Examples

```
# Use a temporary cache so the example doesn't change your cache
old_options = options(annotatr.cache = tempfile())

# Remove the hg19 annotations and downloads
clear_cached_annotations(genome = 'hg19')

# Remove everything
clear_cached_annotations()

options(old_options)
```

---

coannotation_pairs *Function to find the pairs of annotation types on each region*

Description

A region with one annotation gives the pair of that type with itself. A region with more than one annotation gives each pair of its annotations, with the types of each pair in sorted order, so a region annotated to a CpG island and a promoter counts only toward the island / promoter pair, and not toward island / island or promoter / promoter.

Usage

```
coannotation_pairs(tbl)
```

Arguments

tbl	A data.frame of annotated regions, from as.data.frame() on the result of annotate_regions().

Value

A data.frame with columns seqnames, start, end, V1, and V2, with one row per distinct pair of annotation types on each region.

`download_annotation_file` *Function to download a file, with retries, into the cache*

Description

Function to download a file, with retries, into the cache

Usage

```
download_annotation_file(url, genome, cache = TRUE, retries = 3)
```

Arguments

`url`	A string giving the URL.
`genome`	A string giving the genome the file is for, used by `clear_cached_annotations()`.
`cache`	A logical stating whether to use the cache (TRUE) or a temporary file (FALSE).
`retries`	The number of times to try the download.

Value

A string giving the path of the downloaded file. The extension of the URL is kept, so functions that read the file can determine its format.

`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_annotation_rname *Function to get the cache resource name of a built annotation*

Description

The name includes the annotatr version, and for gene annotations, the versions of the `TxDb.*` and `org.*.eg.db` packages, the AnnotationHub EnsDb, or MANE, so updating them causes a rebuild.

Usage

```
get_annotation_rname(code, genome)
```

Arguments

code	A string giving the annotation code, e.g. `'hg19_genes_promoters'`.
genome	A string giving the genome assembly.

Value

A string giving the resource name.

get_bfc *Function to get the BiocFileCache for the cache*

Description

Function to get the BiocFileCache for the cache

Usage

```
get_bfc()
```

Value

A `BiocFileCache` object.

get_cache_dir *Function to get the directory of the cache*

Description

Function to get the directory of the cache

Usage

```
get_cache_dir()
```

Value

A string giving the cache directory, from the `annotatr.cache` option or `tools::R_user_dir()`.

get_cache_rids *Function to get the resource IDs of cache entries*

Description

Function to get the resource IDs of cache entries

Usage

```
get_cache_rids(bfc, rname, exact = TRUE)
```

Arguments

`bfc`	A `BiocFileCache` object.
`rname`	A string giving the resource name.
`exact`	A logical stating whether to match `rname` exactly (TRUE) or as a prefix (FALSE).

Value

A character vector of resource IDs.

`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_chrom_aliases *Function to map any chromosome alias of a genome to its UCSC-style name*

Description

Uses the chromAlias file of the UCSC GenArk hub (for GENARK genomes) or of the UCSC database (goldenPath/[genome]/bigZips), e.g. to rename the Ensembl sequence names of an EnsDb (1, MT, KI270728.1) to UCSC names (chr1, chrM, chr1_KI270728v1_random).

Usage

```
get_chrom_aliases(genome, cache = TRUE)
```

Arguments

genome	A string giving the genome assembly, e.g. 'mm39' or 'oviariramb2'.
cache	A logical stating whether to use the cache on disk for downloads.

Value

A named character vector whose names are aliases (e.g. Ensembl, GenBank, RefSeq, and UCSC names) and whose values are UCSC-style names.

get_genark_seqinfo *Function to get the Seqinfo of a GenArk genome with UCSC-style names*

Description

Function to get the Seqinfo of a GenArk genome with UCSC-style names

Usage

```
get_genark_seqinfo(genome, cache = TRUE)
```

Arguments

genome	A string giving the genome assembly, one of GENARK$genome.
cache	A logical stating whether to use the cache on disk for downloads.

Value

A Seqinfo object.

get_genark_url — *Function to get the URL of a file in the UCSC GenArk hub for a genome*

Description

Function to get the URL of a file in the UCSC GenArk hub for a genome

Usage

```
get_genark_url(genome, file)
```

Arguments

genome	A string giving the genome assembly, one of GENARK$genome.
file	A string giving the path of the file within the hub directory, e.g. 'GCF_016772045.1.chromAlias.txt'.

Value

A string giving the URL.

get_gene_table — *Function to map gene IDs to gene symbols, Entrez IDs, and Ensembl IDs*

Description

When an ID maps to more than one symbol, Entrez ID, or Ensembl ID, the first is used.

Usage

```
get_gene_table(gene_ids, orgdb = NULL, keytype = NULL)
```

Arguments

gene_ids	A character vector of gene IDs.
orgdb	An OrgDb object to map the IDs with, or NULL to only use the IDs themselves (per keytype).
keytype	A string giving the type of gene_ids, as a keytype of orgdb, e.g. 'ENTREZID', 'ENSEMBL', or 'FLYBASE'. Versions of Ensembl IDs are ignored. If NULL, the type is unknown.

Value

A data.frame with one row per unique gene ID, and columns gene_id, symbol, entrez_id, and ensembl_id.

get_mane_summary *Function to get the MANE Select summary*

Description

Function to get the MANE Select summary

Usage

```
get_mane_summary(cache = TRUE)
```

Arguments

cache	A logical stating whether to use the cache on disk for downloads.

Value

A `data.frame` with one row per MANE Select transcript, and columns `tx_id` (Ensembl transcript ID with version), `gene_id` (Entrez ID), `symbol`, and `ensembl_id` (Ensembl gene ID without version).

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.

list_cached_annotations *List cached annotations and downloads*

Description

List the annotations and downloaded files in the cache used by `build_annotations()`. See `cached-annotations` for more about the cache.

Usage

```
list_cached_annotations()
```

Value

A `data.frame` with one row per cached item, and columns `type` (`'annotation'` or `'download'`), `genome`, `name` (the annotation code or URL), `sources` (the versions an annotation was built with), `size_mb`, `added`, and `path`.

Examples

```
# Use a temporary cache so the example doesn't change your cache
old_options = options(annotatr.cache = tempfile())

list_cached_annotations()

options(old_options)
```

load_cached_annotation *Function to load a built annotation from the cache*

Description

A cached annotation that can't be read is removed so it will be rebuilt.

Usage

```
load_cached_annotation(code, genome)
```

Arguments

code	A string giving the annotation code.
genome	A string giving the genome assembly.

Value

A GRanges object, or NULL if the annotation isn't cached.

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 a background of regions, i.e. the regions your data could have come from (e.g. all tested CpGs when annotated_regions are differentially methylated CpGs). Despite the name, this should not be randomized regions; randomize_regions() is deprecated.
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 differentially methylated (DM) regions compared to
 # all regions tested for differential methylation

 # 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')
 tested_regions = read_regions(con = dm_file, genome = 'hg19', extraCols = extraCols,
     rename_score = 'pval', rename_name = 'DM_status', format = 'bed')

 # Annotate all tested regions, which are the background
 tested_annots = annotate_regions(
     regions = tested_regions,
     annotations = annotations,
     ignore.strand = TRUE)

 # The data are the DM regions
 dm_annots = tested_annots[tested_annots$DM_status != 'none']

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

 p_annots = plot_annotation(annotated_regions = dm_annots,
     annotation_order = annots_order)
```

```
p_annots_bg = plot_annotation(annotated_regions = dm_annots,
    annotated_random = tested_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 "Background" bar shows the distribution of the background 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 a background of regions, i.e. the regions your data could have come from (e.g. all tested CpGs when annotated_regions are differentially methylated CpGs). Despite the name, this should not be randomized regions; randomize_regions() is deprecated. A background 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')

# Compare the DM regions to all tested regions, which are the background
dm_sig_annots = dm_annots[dm_annots$DM_status != 'none']

dm_vn_bg = plot_categorical(
    annotated_regions = dm_sig_annots,
    annotated_random = 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')
```

`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 deprecated and will be removed in a future release. Regions placed uniformly at random across the genome are a poor null model for most data, because real regions come from a much narrower set of possible locations (e.g. mappable sequence, or the CpGs an assay measures). Instead, annotate a background of the regions your data could have come from (e.g. all tested CpGs) with `annotate_regions()`, and pass it as `annotated_random` to `summarize_annotations()`, `plot_annotation()`, or `plot_categorical()`. For a statistical test, see `regioneR::permTest()` or `regioneR::resampleRegions()`.

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 each other (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

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().

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.

See Also

annotatr-deprecated

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 = suppressWarnings(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','tx_id','gene_id','symbol','entrez_id','ensembl_id','t
Columns named tx_id, gene_id, symbol, entrez_id, or ensembl_id in extraCols are kept, and the others are NA.

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

The GRanges object of the custom annotations, invisibly. Combine it with other annotations using c(), e.g. c(read_annotations(...), build_annotations(...)). It is also stored in annotatr_cache for the current R session as '[genome]_custom_[name]', which can be included in the call to build_annotations(). See examples below.

Examples

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

# Combine it with other annotations, e.g. the premade CpG annotations
data('annotations', package = 'annotatr')
combined = c(custom, annotations)

# Or refer to it by name in build_annotations()
build_annotations(genome = 'hg19', annotations = 'hg19_custom_test')
```

---

read_ccre_bed *Function to read an ENCODE cCRE registry BED file*

Description

The file has the columns chromosome, start, end, DHS accession, cCRE accession, and class, with 0-based starts as in all BED files.

Usage

```
read_ccre_bed(path, genome)
```

Arguments

path	A string giving the path of the file.
genome	A string giving the genome assembly.

Value

A GRanges object with 1-based coordinates, the genome's seqinfo, and mcols accession and class.

---

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

BED files have 0-based, half-open coordinates, and GRanges are 1-based and closed, so rtracklayer::import() adds 1 to each start. For example, the BED line chr1 99 100 is the single base chr1:100. A file with 1-based starts, e.g. single CpGs written as chr1 100 100, gives ranges of width 0 (chr1:101-100), which overlap no annotations. Such files should have 1 subtracted from their starts first.

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.

---

save_cached_annotation *Function to save a built annotation in the cache*

Description

Any earlier versions of the annotation in the cache are removed.

Usage

```
save_cached_annotation(gr, code, genome)
```

Arguments

gr	A GRanges object of the annotation.
code	A string giving the annotation code.
genome	A string giving the genome assembly.

Value

The path of the cached file, invisibly.

---

set_genome_seqinfo *Function to give a GRanges the seqinfo of a genome*

Description

Some operations, e.g. `rtracklayer::liftOver()`, drop the genome and sequence lengths. This restores them from `Seqinfo::Seqinfo(genome = genome)`. If that fails, e.g. offline, or if the ranges are on sequences the genome doesn't have, only the genome is set.

Usage

```
set_genome_seqinfo(gr, genome)
```

Arguments

gr	A GRanges object.
genome	A string giving the genome assembly, e.g. 'hg38'.

Value

gr with the seqinfo of genome, trimmed to the ends of its sequences.

---

standardize_mcols *Function to give an annotation the standard mcols*

Description

Adds any missing columns of id, tx_id, gene_id, symbol, entrez_id, ensembl_id, and type as NA, and drops any others, so annotations can be combined with c().

Usage

```
standardize_mcols(gr)
```

Arguments

gr	A GRanges object of an annotation.

Value

The GRanges object with the standard mcols, in order.

---

strip_id_version *Function to remove the version from Ensembl IDs*

Description

Function to remove the version from Ensembl IDs

Usage

```
strip_id_version(ids)
```

Arguments

ids	A character vector of IDs, e.g. 'ENSG00000121410.14'.

Value

The IDs without versions, e.g. 'ENSG00000121410'. IDs without a version are unchanged.

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 missing, then the same is computed for the background regions, labeled "Background" in the data_type column.

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 a background of regions, i.e. the regions your data could have come from (e.g. all tested CpGs when annotated_regions are differentially methylated CpGs). Despite the name, this should not be randomized regions; randomize_regions() is deprecated.
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 differentially methylated (DM) regions compared to
### all regions tested for differential methylation

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

file = system.file('extdata', 'IDH2mut_v_NBM_multi_data_chr9.txt.gz', package = 'annotatr')
extraCols = c(diff_meth = 'numeric', mu1 = 'numeric', mu0 = 'numeric')
r = read_regions(con = file, genome = 'hg19', extraCols = extraCols,
    rename_score = 'pval', rename_name = 'DM_status', format = 'bed')

# Annotate all tested regions, which are the background
tested_annots = annotate_regions(
    regions = r,
    annotations = annotations,
    ignore.strand = TRUE,
    quiet = FALSE)

# The data are the DM regions
dm_annots = tested_annots[tested_annots$DM_status != 'none']

# Summarize the annotated DM regions
s = summarize_annotations(annotated_regions = dm_annots)

# Summarize the annotated DM regions and the background
s_bg = summarize_annotations(
    annotated_regions = dm_annots,
    annotated_random = tested_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_genes *Summarize annotated regions by gene*

Description

Given a GRanges of annotated regions, summarize the regions annotated to each gene, with one row per gene. Only gene annotations with a gene ID count (e.g. promoters, 1-5kb upstream, UTRs, exons, and introns), so CpG, intergenic, enhancer, and chromatin annotations are left out.

Usage

```
summarize_genes(
  annotated_regions,
  over = NULL,
  by = NULL,
  format = c("wide", "long"),
  quiet = FALSE
)
```

Arguments

annotated_regions	The GRanges result of annotate_regions(), with gene annotations such as [genome]_basicgenes or hg38_basicmane.
over	A character vector of numerical data columns to summarize with the mean, median, and standard deviation over each gene's regions. Default NULL, no numerical summaries.
by	A single categorical data column to count the categories of over each gene's regions, e.g. 'DM_status'. Default NULL, no category counts.
format	Either 'wide' (the default) for one row per gene with a count column per annotation type, or 'long' for one row per gene and annotation type.
quiet	Print progress messages (FALSE) or not (TRUE).

Details

A region counts once toward a gene, however many of the gene's annotations it overlaps, and once toward each annotation type of the gene. A region annotated to more than one gene counts toward each of them.

MANE annotations (hg38_mane_*) and annotations from build_txdb_annotations() (e.g. mm39_gencode_*) are summarized the same way. With annotations from more than one group, the annotation types of groups other than genes are prefixed with the group, e.g. n_promoters and n_mane_promoters. hg38_genes_* and hg38_mane_* both use Entrez gene IDs, so a gene has one row. Groups with different kinds of gene IDs (e.g. Entrez and Ensembl) give separate rows for the same gene.

Value

A tbl_df with columns gene_id, symbol, entrez_id, and ensembl_id, then for format = 'wide', n_regions and n_[type] for each annotation type (e.g. n_promoters), or for format = 'long', annot.type and n. These are followed by n_[category] for each category in by, and [column]_mean, [column]_median, and [column]_sd for each column in over. Genes with the most regions come first.

Examples

```
if(requireNamespace('TxDb.Hsapiens.UCSC.hg19.knownGene', quietly = TRUE) &&
    requireNamespace('org.Hs.eg.db', quietly = TRUE)) {
 # Build hg19 promoter annotations
 annotations = build_annotations(genome = 'hg19', annotations = 'hg19_genes_promoters')

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

 # One row per gene, with the mean methylation difference and the
 # number of hyper- and hypomethylated regions
genes = summarize_genes(annotated_regions = dm_annots, over = 'diff_meth', by = 'DM_status')

 # One row per gene and annotation type
genes_long = summarize_genes(annotated_regions = dm_annots, over = 'diff_meth', format = 'long')
}
```

---

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.

`ucsc_seqlevels` *Function to rename sequences to UCSC-style names*

Description

Renames sequences from any alias in the genome's chromAlias file (see `get_chrom_aliases()`), e.g. the Ensembl names of an `EnsDb`, to UCSC-style names, and sets the genome's full `seqinfo`. Sequences without a UCSC name are dropped.

Usage

```
ucsc_seqlevels(gr, genome, cache = TRUE)
```

Arguments

`gr`	A `GRanges` or `GRangesList`.
`genome`	A string giving the genome assembly, e.g. `'mm39'` or `'oviariramb2'`.
`cache`	A logical stating whether to use the cache on disk for downloads.

Value

`gr` with UCSC-style sequence names and the genome's `seqinfo`.

Index

∗ **datasets**
    annotations, 5
    annotatr_cache, 6

annotate_regions, 3
annotations, 5
annotatr, 5
annotatr-package *(annotatr)*, 5
annotatr_cache, 6

build_ah_annots, 7
build_annotations, 7, *14*, *18*
build_ccre_annots, 9
build_cpg_annots, 10
build_enhancer_annots, 10
build_gene_annots, 11
build_hmm_annots, 12
build_lncrna_annots, 12
build_mane_txdb, 13
build_txdb_annotations, 13
build_txdb_gene_annots, 15
builtin_annotations, 15
builtin_annotations_table, 16
builtin_genomes, 17

cached-annotations, 17
check_annotations, 18
check_regions_genome, 19
clear_cached_annotations, *18*, 19
coannotation_pairs, 20

download_annotation_file, 21

expand_annotations, 21

get_annotation_rname, 22
get_bfc, 22
get_cache_dir, 23
get_cache_rids, 23
get_cellline_from_code, 24
get_cellline_from_shortcut, 24
get_chrom_aliases, 25
get_genark_seqinfo, 25
get_genark_url, 26
get_gene_table, 26
get_mane_summary, 27
get_orgdb_name, 27
get_txdb_name, 28

list_cached_annotations, *18*, 28
load_cached_annotation, 29

plot_annotation, 29
plot_categorical, 31
plot_coannotations, 33
plot_numerical, 35
plot_numerical_coannotations, 37

randomize_regions, 39
read_annotations, *14*, 40
read_ccre_bed, 41
read_regions, 42
reformat_hmm_codes, 43

save_cached_annotation, 44
set_genome_seqinfo, 44
standardize_mcols, 45
strip_id_version, 45
subset_order_tbl, 46
summarize_annotations, 46
summarize_categorical, 47
summarize_genes, 49
summarize_numerical, 50

tidy_annotations, 52

ucsc_seqlevels, 52