Package 'epiRomics'

June 5, 2026

Type Package

Title Epigenomic Analysis Package Built for R (epiRomics)

Version 1.1.0

Description Integrates various levels of epigenomic information, including ChIP-seq, histone modification, ATAC-seq, and RNA-seq data. Regulatory network analysis uses combinatory approaches to infer regions of significance, such as enhancers. Downstream analysis identifies co-occurrence of epigenomic data at regions of interest. Visualization functions display multi-track genomic views with signal overlays. Please contact <ammawla@ucdavis.edu> for suggestions, feedback, or bug reporting.

License Artistic-2.0

Depends R (>= 4.5.0)

URL https://huising-lab.github.io/epiRomics/, https://github.com/Huising-Lab/epiRomics

BugReports https://github.com/Huising-Lab/epiRomics/issues

biocViews Epigenetics, ChIPSeq, ATACSeq, RNASeq, Visualization, Sequencing, Software, HistoneModification, GeneRegulation, Transcription, FunctionalGenomics

Imports AnnotationDbi (>= 1.68.0), annotatr (>= 1.32.0), BiocGenerics (>= 0.52.0), ChIPseeker (>= 1.42.0), data.table (>= 1.15.0), digest (>= 0.6.35), GenomeInfoDb (>= 1.42.0), GenomicFeatures (>= 1.58.0), GenomicRanges (>= 1.58.0), graphics, grDevices, IRanges (>= 2.40.0), methods, rtracklayer (>= 1.66.0), S4Vectors (>= 0.44.0), stats, tools, utils

Suggests BiocFileCache (>= 2.14.0), knitr, org.Hs.eg.db (>= 3.20.0), org.Mm.eg.db (>= 3.20.0), parallel, rmarkdown, testthat (>= 3.0.0), TxDb.Hsapiens.UCSC.hg38.knownGene (>= 3.18.0), TxDb.Mmusculus.UCSC.mm10.knownGene (>= 3.10.0)

Config/testthat/edition 3

ByteCompile true

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.3

Collate 'class-epiRomics.R' 'accessors.R'
'benchmark_enhancer_predictor.R' 'build_database.R' 'cache.R'
'chromatin_states.R' 'enhanceosomes.R' 'enhancers.R'
'epiRomics-package.R' 'filter_enhancers.R' 'plot_quick_view.R'
'plot_tracks_fast.R' 'predictors.R' 'putative_enhancers.R'
'regions_of_interest.R' 'synthetic-data.R' 'tf_overlap.R'
'utils.R' 'visualizations.R' 'zzz-deprecated.R' 'zzz.R'

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

git_branch devel

git_last_commit 9e2708e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Alex M. Mawla [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0907-464X>),
Mark O. Huising [aut] (ORCID: <https://orcid.org/0000-0002-6594-2205>)

Maintainer Alex M. Mawla <ammawla@ucdavis.edu>

Contents

`epiRomics-package` *epiRomics: Epigenomic Analysis Package Built for R (epiRomics)*

Description

Integrates various levels of epigenomic information, including ChIP-seq, histone modification, ATAC-seq, and RNA-seq data. Regulatory network analysis uses combinatory approaches to infer regions of significance, such as enhancers. Downstream analysis identifies co-occurrence of epigenomic data at regions of interest. Visualization functions display multi-track genomic views with signal overlays. Please contact <ammawla@ucdavis.edu> for suggestions, feedback, or bug reporting.

Author(s)

Maintainer: Alex M. Mawla <ammawla@ucdavis.edu> (ORCID)

Authors:

- Mark O. Huising <mhuising@ucdavis.edu> (ORCID)

See Also

Useful links:

- https://huising-lab.github.io/epiRomics/
- https://github.com/Huising-Lab/epiRomics
- Report bugs at https://github.com/Huising-Lab/epiRomics/issues

`analyze_tf_cobinding` *Analyze statistical significance of TF co-binding at enhanceosome regions*

Description

Performs pairwise statistical testing of transcription factor co-occurrence at enhanceosome regions using Fisher's exact test or permutation testing, with odds ratios, Pointwise Mutual Information (PMI), and hierarchical clustering.

Usage

```
analyze_tf_cobinding(
  enhanceosome,
  database,
  fdr_threshold = 0.05,
  min_regions = 5L,
  method = c("fisher", "permutation"),
  n_permutations = 1000L
)
```

Arguments

`enhanceosome`	epiRomics class database containing enhanceosome calls
`database`	epiRomics class database containing all data initially loaded
`fdr_threshold`	numeric, FDR threshold for significance (default: 0.05)
`min_regions`	integer, minimum number of co-bound regions to report a pair (default: 5)
`method`	character, statistical method: "fisher" (default) for Fisher's exact test or "permutation" for permutation-based testing that accounts for spatial autocorrelation.
`n_permutations`	integer, number of permutations when `method = "permutation"` (default: 1000). Ignored for Fisher's test.

Details

This replaces the previous decision-tree approach (`epiRomics_predictors`) with statistically rigorous co-binding analysis. For each pair of TFs, a 2x2 contingency table is constructed from the enhanceosome presence matrix. P-values are corrected using Benjamini-Hochberg FDR.

Value

list with components:

pairwise data.frame with columns: tf1, tf2, n_both, n_tf1_only, n_tf2_only, n_neither, odds_ratio, pvalue, fdr, pmi, significant

presence_matrix logical matrix (regions x TFs) of binding presence

clustering hclust object from hierarchical clustering of TF co-occurrence (Jaccard distance, Ward.D2 linkage)

tf_names character vector of TF names analyzed

n_regions integer, total number of enhanceosome regions

method character, statistical method used

Statistical methods

- **Fisher's exact test** (method = "fisher"): Tests whether two TFs co-occur at enhanceosome regions more (or less) often than expected by chance. Assumes independence between regions. This is the default and is appropriate when regions are largely non-overlapping. Reference: Fisher, R.A. (1922) J Royal Stat Soc.
- **Permutation test** (method = "permutation"): Shuffles TF_B binding labels across regions to generate a null distribution, accounting for spatial autocorrelation between nearby genomic regions. More conservative but robust to violations of independence. Reference: Gel et al. (2016) Bioinformatics 32(2):289-291. "regioneR: an R/Bioconductor package for the association analysis of genomic regions."
- **Odds ratio**: Measures strength of association. OR > 1 indicates co-occurrence; OR < 1 indicates mutual exclusion.
- **PMI**: Pointwise Mutual Information quantifies the degree of association between two TFs: `PMI(A,B) = log2(P(A,B) / (P(A)*P(B)))`. PMI > 0 indicates co-occurrence; PMI < 0 indicates avoidance. Reference: Church & Hanks (1990) Computational Linguistics.
- **BH-FDR**: Benjamini-Hochberg correction controls the false discovery rate across all pairwise tests. Reference: Benjamini & Hochberg (1995) J Royal Stat Soc B.

Note on spatial autocorrelation

Fisher's exact test assumes independence between observations (regions). Nearby genomic regions may be spatially correlated (e.g., broad TF binding domains), which can inflate significance. If your enhanceosome regions contain many closely spaced or overlapping intervals, consider using `method = "permutation"` for more conservative p-values.

See Also

`analyze_tf_overlap` for overlap fractions without significance testing

Examples

```
db <- make_example_database()
eso <- make_example_enhanceosome(db)
cobinding <- analyze_tf_cobinding(eso, db)
cobinding$pairwise[, c("tf1", "tf2", "odds_ratio", "fdr")]
```

---

analyze_tf_overlap *Analyze pairwise and multi-way overlap between transcription factor binding sites*

Description

For N TFs in the enhanceosome, computes pairwise overlap fractions, unique region counts per TF, shared region counts, and UpSet-style intersection data for all TF combinations. Uses Jaccard index for symmetric overlap quantification and overlap coefficient for asymmetric assessment (Church & Hanks, 1990). The presence/absence matrix pattern follows the approach used by DiffBind (Stark & Brown, 2011).

Usage

```
analyze_tf_overlap(enhanceosome, database, regions = NULL)
```

Arguments

enhanceosome	epiRomics class database containing enhanceosome calls, or an epiRomics database with ChIP data
database	epiRomics class database containing all data initially loaded
regions	GRanges object defining regions to analyze. If NULL, uses all enhanceosome annotation regions.

Value

list with components:

overlap_matrix matrix of pairwise overlap coefficients (asymmetric: fraction of row TF overlapping col TF)

overlap_counts matrix of pairwise absolute overlap counts

jaccard_matrix matrix of pairwise Jaccard indices (symmetric: intersection/union)

unique_counts named integer vector of regions bound by ONLY that TF

shared_all integer count of regions bound by ALL TFs

n_tf_counts table of how many regions are bound by exactly 1, 2, 3... N TFs

tf_names character vector of TF names analyzed

summary data.frame with per-TF summary statistics

References

- Church KW, Hanks P (1990) Computational Linguistics 16(1):22-29. "Word Association Norms, Mutual Information, and Lexicography." Pointwise mutual information framework adapted for co-binding analysis.
- Stark R, Brown GD (2011) DiffBind, Bioconductor. Presence/absence matrix approach for binding site overlap analysis.

Examples

```
db <- make_example_database()
eso <- make_example_enhanceosome(db)
overlap <- analyze_tf_overlap(eso, db)
overlap$overlap_matrix
```

---

annotate_enhancers *Annotate putative enhancers with functional database overlaps*

Description

Cross-references putative enhancers against all functional databases (FANTOM5, UCNEs, Regulome Active, Regulome Super, etc.) loaded in the epiRomics database. Adds boolean overlap columns for each database, a count of overlapping databases, and a `novel` flag for enhancers not found in any known database.

Usage

```
annotate_enhancers(putative_enhancers, database)
```

Arguments

putative_enhancers	data.frame. Output from find_putative_enhancers.
database	An epiRomics S4 database object.

Value

The input data.frame with additional columns:

overlap_<name> Logical. TRUE if putative enhancer overlaps the named functional database (one column per database).

n_databases Integer. Count of functional databases overlapping.

novel Logical. TRUE if no functional database overlap (completely novel enhancer call).

Examples

```
db <- make_example_database()
pe <- find_putative_enhancers(db)
pe_annot <- annotate_enhancers(pe, db)
table(pe_annot$novel)
```

---

annotations *Access the genomic annotations slot of an epiRomicsS4 object*

Description

Returns the GRanges object stored in the annotations slot. Public accessor replacement for obj@annotations or methods::slot(obj, "annotations").

Usage

```
annotations(x, ...)

## S4 method for signature 'epiRomicsS4'
annotations(x, ...)
```

Arguments

x	An epiRomicsS4 object.
...	Currently unused; reserved for method extension.

Value

A GRanges object. Empty (length 0) if no annotations have been assigned.

See Also

annotations<-, epiRomicsS4

Examples

```
db <- make_example_database()
annotations(db)
length(annotations(db))
```

`annotations-set` *Replace the annotations slot of an epiRomicsS4 object*

Description

Assigns a new `GRanges` object to the `annotations` slot. Validity is checked via `validObject`.

Usage

```
annotations(x, ...) <- value

## S4 replacement method for signature 'epiRomicsS4'
annotations(x, ...) <- value
```

Arguments

`x`	An `epiRomicsS4` object.
`...`	Currently unused; reserved for method extension.
`value`	A `GRanges` object.

Value

The updated `epiRomicsS4` object.

Examples

```
db <- make_example_database()
gr <- GenomicRanges::GRanges("chr1", IRanges::IRanges(1, 100))
gr$type <- "hg38_custom_h3k4me1"
annotations(db) <- gr
length(annotations(db))
```

`benchmark_enhancer_predictor`
Evaluate histone marks against any curated reference database

---

Description

Returns overlap fractions per histone mark against a reference database, allowing comparison of which marks best predict regions in the reference. Supports any database in database, not just FANTOM.

Usage

```
benchmark_enhancer_predictor(
  database,
  histone = "h3k4me1",
  curated_database = "fantom"
)
```

Arguments

database	epiRomics class database containing all data initially loaded
histone	name or vector of histone mark(s), must match a name entry in meta(database), default set to h3k4me1
curated_database	database to test histone marks against, must match a name entry in meta(database), default set to fantom

Value

Variable of class dataframe further exploring top histone marks that may determine enhancer regions

Examples

```
db <- make_example_database()
## Use h3k27ac as the curated reference; compare h3k4me1 against it
test <- benchmark_enhancer_predictor(
  db,
  histone = "h3k4me1",
  curated_database = "h3k27ac"
)
test
```

---

build_database *Build epiRomics database*

Description

Reads epigenomic annotation files from a CSV manifest and builds a unified epiRomics database for downstream analysis. Supports optional extra columns for ChIP/histone peak files (signal, pval, qval, peak).

Usage

```
build_database(
  db_file,
  txdb_organism,
  genome,
  organism,
  extraCols = NULL,
  data_dir = NULL
)
```

Arguments

db_file	character string of path to properly formatted csv file containing epigenetic data. [See vignette for more details]
txdb_organism	a character string containing the TxDB associated with your data.
genome	a character string naming the genome assembly associated with your data (e.g. "mm10", "hg38", "rn6", "dm6"). The value must match the assembly referenced by txdb_organism / the CSV manifest's genome column; epiRomics itself is organism-agnostic.
organism	a character string containing the org.db associated with your data.
extraCols	named character vector of extra columns to read from chip/histone BED files. Default is NULL (no extra columns). Set to c(signal = "numeric", pval = "numeric",qval = "numeric", peak = "numeric") to read narrowPeak columns.
data_dir	optional character string specifying the root directory for resolving relative file paths in the CSV manifest. When provided, any relative path in the path column is prefixed with data_dir. This is especially useful with cached data from cache_data where the CSV uses relative paths. Default is NULL (paths used as-is).

Value

Variable of class epiRomics for further downstream analysis

Examples

```
## build_database reads external BED/BigWig files from a CSV manifest.
## Confirm that a missing file produces a clean error:
tryCatch(
  build_database("nonexistent.csv",
    txdb_organism = paste0("TxDb.Hsapiens.UCSC.hg38.knownGene::",
                           "TxDb.Hsapiens.UCSC.hg38.knownGene"),
    genome = "hg38", organism = "org.Hs.eg.db"),
  error = function(e) message(e$message)
)
```

---

cache_data *Download and Cache epiRomics Example Data*

Description

Downloads the epiRomics example dataset (histone marks, ChIP-seq peaks, BED annotations, BigWig signal files, and differential analysis results) from a remote archive and caches it locally using **BiocFileCache**. Subsequent calls return the cached path without re-downloading.

Usage

```
cache_data(force_update = FALSE, ask = FALSE)
```

Arguments

force_update	Logical; if TRUE, re-download the archive even when a cached copy exists.
ask	Logical; passed to BiocFileCache::BiocFileCache(). Set to FALSE (default) for non-interactive use.

Details

The example dataset is approximately 1.3 GB compressed and includes:

Histone/ H3K27ac, H3K4me1, H3K27me3, H3K9me3, H3K4me3, H3K36me3, and H2A.Z peak calls (BED format)

ChIP/ Transcription factor peak calls for FOXA2, MAFB, NKX2.2, NKX6.1, and PDX1 (BED format)

BED_Annotation/ FANTOM5 enhancers, UCNEs, and Human Islet Regulome active/super enhancers (BED format)

BigWigs/ ATAC-seq and RNA-seq signal tracks for human pancreatic islet alpha and beta cells (bigWig format)

CSV files DiffBind differential accessibility results and RNA-seq differential expression results

Data are downloaded once and stored in the BiocFileCache directory (typically `~/.cache/R/BiocFileCache` on Linux/macOS).

Value

A character string giving the path to the local directory containing the example data files (BigWigs, BED annotations, ChIP peaks, histones, CSV files, and template sheets).

Internet requirement

The first call requires internet access to download the data archive. Subsequent calls work offline using the local cache. Use has_cache to test whether the data is already available before attempting to build vignettes or run examples.

See Also

has_cache to check data availability without triggering a download.

Examples

```
## Check whether cached data is already available (no network).
has_cache()

## A lightweight toy subset is always bundled with the package and
## does not require cache_data(). Use it for quick demos:
toy_dir <- system.file("extdata", "toy", package = "epiRomics")
list.files(toy_dir)


## Network download (~1.3 GB). Wrapped in \donttest{} per Bioconductor
## guidance. Only run interactively:
if (interactive()) {
  cache_dir <- cache_data()
}
```

`call_accessible_regions` *Compute per-region signal z-scores from a BigWig file*

Description

Calculates the mean ATAC/ChIP signal per region from a BigWig file and returns z-scores relative to the distribution across all input regions. The null distribution is estimated from the signal across all regions, assuming most of the genome has low/no signal. Regions with z-scores above the threshold are called as accessible.

Usage

```
call_accessible_regions(
  bw_path,
  regions,
  z_threshold = 1,
  auto_threshold = FALSE,
  return_scores = FALSE
)
```

Arguments

`bw_path`	character, path to a BigWig file.
`regions`	GRanges object of regions to classify.
`z_threshold`	numeric, z-score cutoff for calling accessibility (default: 1.0). Regions with z >= threshold are called accessible. Set to NULL to return z-scores without thresholding. Ignored when `auto_threshold = TRUE`.
`auto_threshold`	logical. If TRUE, automatically detects the optimal threshold by finding the valley between the noise and signal peaks in the log-transformed signal distribution using kernel density estimation. This is the recommended approach – let the data define background vs. signal rather than using a fixed z-score (default: FALSE).
`return_scores`	logical. If TRUE, returns a list with z_scores, raw signal, and boolean calls. If FALSE (default), returns a logical vector for backward compatibility.

Value

If `return_scores = FALSE` (default): a logical vector the same length as `regions`, TRUE for accessible regions. If `return_scores = TRUE`: a list with:

z_scores numeric vector of z-scores

signal numeric vector of raw mean signal per region

accessible logical vector (z >= threshold)

mu mean signal (background estimate)

sigma SD of signal

Examples

```
bw_path <- make_example_bigwig()
regions <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(
    start = c(1000L, 5000L, 10000L, 20000L, 50000L),
    end   = c(2000L, 6000L, 11000L, 21000L, 51000L)
  )
)
accessible <- call_accessible_regions(bw_path, regions, z_threshold = 1)
file.remove(bw_path)
```

---

chromatin_state_categories

Return chromatin state category definitions

Description

Variant of classify_chromatin_states that outputs simplified single-state categories. Resolves dual-mark states (bivalent = H3K4me3 + H3K27me3, poised = H3K4me1 + H3K27me3) to the single most likely state, useful for applications that require one-state-per-region classification.

Usage

```
chromatin_state_categories(
  database,
  histone_marks = NULL,
  regions = NULL,
  refine_by_tss = TRUE,
  tss_window = 2000L
)
```

Arguments

database	epiRomics class database containing all data initially loaded
histone_marks	character vector of histone mark names to use for classification. Must match names in meta(database). If NULL, auto-detects from meta.
regions	GRanges object of regions to classify. If NULL, uses all annotations in the database.
refine_by_tss	logical. If TRUE (default), promoter states are assigned only to regions within tss_window of an annotated TSS. Regions with promoter marks (H3K4me3) outside TSS windows are reclassified as enhancers.
tss_window	integer. Distance in bp around each TSS to define the promoter zone (default: 2000L). Regions within +/- tss_window of any annotated TSS are considered "promoter" context.

Details

State resolution rules:

- **active_promoter**: H3K4me3 + H3K27ac (replaces "active" when K4me3 present; includes former bivalent with K27ac)

- **active_enhancer**: H3K4me1 + H3K27ac (distal active elements)
- **active**: H3K27ac alone (sufficient per Creyghton et al. 2010)
- **primed_enhancer**: H3K4me1 alone (no K27ac, no K27me3)
- **transcribed**: H3K36me3 alone (SETD2/Pol II gene body mark; standard epiRomics marks this as "unmarked")
- **polycomb_repressed**: H3K27me3 (Polycomb Repressive Complex; absorbs former poised and bivalent-without-K27ac)
- **heterochromatin**: H3K9me3 (constitutive heterochromatin)
- **quiescent**: No histone marks detected
- **low_signal**: Borderline marks or H2A.Z only

Bivalent resolution: K4me3 + K27me3 + K27ac -> active_promoter (K27ac sufficient for active). K4me3 + K27me3 without K27ac -> polycomb_repressed (K27me3 dominates without activating mark).

Poised resolution: K4me1 + K27me3 -> polycomb_repressed (K27me3 dominates in single-state HMM framework).

Value

data.frame with same structure as `classify_chromatin_states` but chromatin_state uses ChromHMM-compatible labels: active_promoter, active_enhancer, active, primed_enhancer, transcribed, polycomb_repressed, heterochromatin, quiescent, low_signal

References

Ernst J, Kellis M (2012) Nature Methods 9(3):215-216. "ChromHMM: automating chromatin-state discovery and characterization."

See Also

`classify_chromatin_states` for the standard epiRomics classification with bivalent/poised states.

Examples

```
db <- make_example_database()
cats <- chromatin_state_categories(db)
table(cats$chromatin_state)
```

---

classify_celltype_accessibility

Classify regions by cell-type-specific binary accessibility

Description

For each genomic region, determines whether it is accessible in each cell type using per-sample z-score thresholding on BigWig ATAC-seq signal. Unlike fold-change approaches (which compare relative enrichment between samples), this method makes an independent binary open/closed call per sample, then classifies regions based on the combination of binary calls.

Usage

```
classify_celltype_accessibility(
  bw_paths,
  regions,
  z_threshold = 1,
  auto_threshold = FALSE
)
```

Arguments

bw_paths	named character vector of BigWig file paths. Names are cell-type labels (e.g., c(Alpha = "alpha.atac.bw",Beta = "beta.atac.bw")).
regions	GRanges object of regions to classify.
z_threshold	numeric, z-score cutoff for calling a region as accessible (default: 1.0). Higher values are more stringent. Use plot_signal_histogram to visualize the signal distribution and identify an appropriate threshold for your data.
auto_threshold	logical. If TRUE, automatically detect the optimal threshold from the signal distribution using kernel density valley detection, overriding z_threshold (default: FALSE).

Details

A region with high signal in BOTH cell types is correctly classified as "Shared" regardless of the magnitude difference between them. This avoids the fold-change pitfall where a region with signal 50 in one cell type and 100 in the other is called "enriched" even though both have accessible chromatin.

Value

A data.frame with columns:

region_idx integer, index into input regions

cell_type character, cell-type label (single name if accessible in only one cell type), "Shared" if accessible in all cell types, or "Closed" if not accessible in any

accessible_in character, comma-separated list of cell types where the region is accessible (empty string if closed)

The binary accessibility matrix (logical, regions x cell types) is attached as the "accessibility_matrix" attribute.

Methodology

For each BigWig file independently:

1. Import mean ATAC signal per region
2. Compute genome-wide null distribution (mean and SD of all region signals)
3. Apply z-score threshold: z = (signal - mean) / sd
4. Regions with z >= threshold are called accessible (open)

The z-score approach normalizes within each sample, making the open/closed call robust to differences in sequencing depth or library complexity between samples (Buenrostro et al. 2013, *Nat Methods*; Corces et al. 2018, *Science*).

Examples

```
bw_path <- make_example_bigwig()
regions <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(
    start = c(1000L, 5000L, 10000L, 20000L, 50000L),
    end   = c(2000L, 6000L, 11000L, 21000L, 51000L)
  )
)
ct <- classify_celltype_accessibility(
  bw_paths = c(Alpha = bw_path, Beta = bw_path),
  regions  = regions
)
table(ct$cell_type)
file.remove(bw_path)
```

---

classify_chromatin_states

Classify genomic regions by histone chromatin state with genomic context

---

Description

Given histone mark combinations in the epiRomics database, classifies regions based on curated chromatin state definitions (ChromHMM/Roadmap Epigenomics conventions). States are refined by TSS proximity so that "promoter" labels are assigned only to regions near transcription start sites (within `tss_window` bp). Regions with promoter-associated marks (H3K4me3) that fall outside TSS windows are reclassified as enhancers (e.g., "active_enhancer" instead of "active_promoter").

Usage

```
classify_chromatin_states(
  database,
  histone_marks = NULL,
  regions = NULL,
  refine_by_tss = TRUE,
  tss_window = 2000L
)
```

Arguments

`database`	epiRomics class database containing all data initially loaded
`histone_marks`	character vector of histone mark names to use for classification. Must match names in `meta(database)`. If NULL, auto-detects from meta.
`regions`	GRanges object of regions to classify. If NULL, uses all annotations in the database.
`refine_by_tss`	logical. If TRUE (default), promoter states are assigned only to regions within `tss_window` of an annotated TSS. Regions with promoter marks (H3K4me3) outside TSS windows are reclassified as enhancers.
`tss_window`	integer. Distance in bp around each TSS to define the promoter zone (default: 2000L). Regions within +/- `tss_window` of any annotated TSS are considered "promoter" context.

Details

Chromatin state definitions (6 simplified labels, priority order):

- **repressed**: H3K27me3 + H3K9me3, or H3K9me3 alone, or H3K27me3 alone (Polycomb/heterochromatin)
- **bivalent**: H3K4me3 + H3K27me3 (poised for activation)
- **active**: H3K4me3 + H3K27ac; H3K4me1 + H3K27ac; H3K4me1 + H3K27ac + H3K36me3; H3K36me3 alone; H2A.Z + H3K27ac
- **poised**: H3K4me1 + H3K27me3; H2A.Z + H3K27me3
- **primed**: H3K4me1 only
- **unmarked**: no marks, H2A.Z alone, or unclassifiable

Genomic context (promoter/gene_body/intergenic) is reported separately so users can infer regulatory identity from position.

Value

data.frame with columns: seqnames, start, end, chromatin_state, genomic_context ("promoter"/"gene_body"/"intergenic"
marks_present (comma-separated), n_marks, is_hotspot

TSS Refinement

H3K4me3 is a promoter-specific mark that peaks at TSS regions (Santos-Rosa et al. 2002, Bernstein et al. 2005). When H3K4me3-containing states are observed outside TSS windows, they likely represent either: (a) strong/broad enhancers that recruit H3K4me3 (Pekowska et al. 2011), or (b) unannotated alternative promoters. By default, these are reclassified as enhancer states. Set `refine_by_tss = FALSE` to disable this behavior.

Genomic Context

A `genomic_context` column is added to the output indicating whether each region is at a "promoter" (within `tss_window` of a TSS), "gene_body" (overlapping a gene but not near TSS), or "intergenic" (not overlapping any annotated gene).

References

- Ernst J, Kellis M (2012) Nature Methods 9(3):215-216. "ChromHMM: automating chromatin-state discovery and characterization."
- Kundaje A et al. (2015) Nature 518(7539):317-330. "Integrative analysis of 111 reference human epigenomes."
- Creyghton MP et al. (2010) PNAS 107(50):21931-21936. "Histone H3K27ac separates active from poised enhancers."
- Rada-Iglesias A et al. (2011) Nature 470(7333):279-283. "A unique chromatin signature uncovers early developmental enhancers in humans."
- Santos-Rosa H et al. (2002) Nature 419(6905):407-411. "Active genes are tri-methylated at K4 of histone H3." H3K4me3 TSS specificity.
- Pekowska A et al. (2011) EMBO J 30(20):4198-4210. H3K4me3 at strong enhancers.

Examples

```
db <- make_example_database()
states <- classify_chromatin_states(db)
table(states$chromatin_state)
```

epiRomics-deprecated *Renamed epiRomics functions*

Description

These functions have been renamed in epiRomics 0.99.1 to follow Bioconductor naming conventions. They remain as thin wrappers that emit a one-time rename notice and forward to the new name. They will be removed in the next release.

Usage

```
epiRomics_build_dB(...)

epiRomics_cache_data(...)

epiRomics_cache_path(...)

epiRomics_has_cache(...)

epiRomics_chromatin_states(...)

epiRomics_chromatin_states_categories(...)

epiRomics_enhanceosome(...)

epiRomics_enhancer_predictor_to_ref(...)

epiRomics_enhancers_co_marks(...)

epiRomics_enhancers_filter(...)

epiRomics_filter_accessible(...)

epiRomics_annotate_putative(...)

epiRomics_putative_enhancers(...)

epiRomics_quick_view(...)

epiRomics_regions_of_interest(...)

epiRomics_tf_cobinding(...)

epiRomics_tf_overlap(...)

epiRomics_track_layer(...)

epiRomics_track_layer_fast(...)

epiRomics_track_layer_gene(...)
```

Arguments

...	Arguments forwarded to the replacement function.

`epiRomicsS4-accessors` *Accessors for the epiRomicsS4 class*

Description

The `epiRomicsS4` class has five slots. Every slot is exposed through a public getter and setter method so users should never need to reach into the object with `obj@slot` or `methods::slot(obj, "slot")`.

Details

Slot	Getter	Setter
`annotations`	`annotations(x)`	`annotations(x) <- value`
`meta`	`meta(x)`	`meta(x) <- value`
`txdb`	`txdb(x)`	`txdb(x) <- value`
`organism`	`organism(object)`	`organism(object) <- value`
`genome`	`genome(x)`	`genome(x) <- value`

`organism()` extends the generic from **BiocGenerics** and `genome()` extends the generic from **GenomeInfoDb**, so `epiRomicsS4` objects respond to those accessors the same way other Bioconductor objects do. `annotations()`, `meta()`, and `txdb()` are generics scoped to **epiRomics**.

Every setter validates the updated object via `validObject` before returning it, so invalid assignments (for example an empty-string `genome`) fail fast with an informative error.

Value

An overview topic; see the individual accessor pages for call signatures and return values.

See Also

`epiRomicsS4`

Examples

```
db <- make_example_database()

# Read every slot through its getter
annotations(db)
meta(db)
txdb(db)
organism(db)
genome(db)

# Setters return an updated object
db2 <- db
genome(db2) <- "mm10"
organism(db2) <- "org.Mm.eg.db"
genome(db2)
organism(db2)
```

epiRomicsS4-class *An S4 class to manage epiRomics databases and downstream results*

Description

The epiRomicsS4 class holds a genomic annotation set along with the metadata needed to interpret it (data-source catalog, TxDb identifier, organism annotation package, genome assembly). Every slot is exposed through a public getter and setter — users should access slot contents through those accessors rather than obj@slot or methods::slot(obj, "slot").

Value

An S4 class definition for epiRomicsS4

Slots

annotations GRanges object containing genomic annotations. Access via annotations().

meta data.frame containing metadata about loaded data sources. Access via meta().

txdb character string of TxDb package::object name. Access via txdb().

organism character string of org.db package name. Access via organism().

genome character string of genome assembly name (e.g., 'mm10', 'hg38'). Access via genome().

See Also

Public accessors for this class: annotations, annotations<-, meta, meta<-, txdb, txdb<-, organism, organism<-, genome, genome<-. Overview: epiRomicsS4-accessors.

Examples

```
showClass("epiRomicsS4")
db <- make_example_database()
genome(db)
organism(db)
length(annotations(db))
```

filter_accessible_regions *Filter putative enhancers by chromatin accessibility evidence*

Description

Multi-mode accessibility filter for putative enhancers. Supports four complementary evidence types that can be used independently or combined.

Usage

```
filter_accessible_regions(
  putative_enhancers,
  track_connection = NULL,
  mode = "signal",
  scope = "filter_distal",
  signal_threshold = 2,
  bed_path = NULL,
  gene_list = NULL,
  promoter_distance = 2000L
)
```

Arguments

putative_enhancers	data.frame. Output from find_putative_enhancers. Must contain columns chr, start, end. For genelist mode, also requires a SYMBOL column.
track_connection	data.frame or NULL. BigWig track connection sheet. Required for mode = "signal".
mode	Character. Filtering mode: "signal" (default), "bed", "genelist", or "combined".
scope	Character. Filtering scope: "filter_distal" (default) or "filter_all".
signal_threshold	Numeric. Z-score threshold for signal mode (default 2).
bed_path	Character or NULL. Path to a BED file for bed mode.
gene_list	Character vector or NULL. Expressed gene symbols for genelist mode.
promoter_distance	Integer. Distance from TSS to classify as promoter-proximal (default 2000 bp). Only used when scope = "filter_distal".

Value

The input data.frame with an atac_accessible logical column. For signal mode, also includes per-sample signal and accessibility columns.

Modes

signal Import ATAC-seq/DNase-seq BigWig signal over each region. Regions with mean signal above a z-score threshold are flagged accessible. Requires track_connection with ATAC/DNase tracks.

bed Overlap with an external accessibility BED file (e.g., ENCODE peaks, DHS hotspots). Any region overlapping a BED entry is flagged. Requires bed_path.

genelist Retain enhancers linked to expressed genes. Regions whose SYMBOL column matches a gene in gene_list are flagged. Requires gene_list.

combined Union of all available evidence. A region is retained if ANY mode flags it as accessible.

Scope

The scope parameter controls which regions are evaluated:

filter_distal Only distal (non-promoter) enhancers are filtered; promoter-proximal regions are always retained. (Default)

filter_all All regions are subject to filtering, including promoter-proximal ones.

Examples

```
db <- make_example_database()
pe <- find_putative_enhancers(db)
## Gene-list mode: attach synthetic SYMBOL column, then filter
pe$SYMBOL <- paste0("GENE", seq_len(nrow(pe)))
pe_genes <- filter_accessible_regions(
  pe, mode = "genelist", gene_list = c("GENE1", "GENE2")
)
sum(pe_genes$atac_accessible)
```

`filter_enhancers` *Filters putative enhancers called by epiRomics_enhancers by crossing against curated FANTOM data*

Description

Filters putative enhancers called by epiRomics_enhancers by crossing against curated FANTOM data

Usage

```
filter_enhancers(putative_enhancers, database, type = NULL)
```

Arguments

`putative_enhancers`	epiRomics class database containing putative enhancer calls
`database`	epiRomics class database containing all data initially loaded
`type`	epiRomics reference containing database to validate putative enhancers against. Default NULL derives from genome: `paste0(genome, "_custom_fantom")`.

Value

Variable of class epiRomics with filtered candidate enhancer regions

Examples

```
db <- make_example_database()
eso <- make_example_enhanceosome(db)
## filter_enhancers requires fantom annotations in database
tryCatch(
  filter_enhancers(eso, db),
  error = function(e) message(e$message)
)
```

`find_enhanceosomes` *Identifies putative enhanceosome regions by cross-referencing candidate enhancer regions against co-TF enrichment*

Description

Identifies putative enhanceosome regions by cross-referencing candidate enhancer regions against co-TF enrichment

Usage

```
find_enhanceosomes(putative_enhancers, database)
```

Arguments

`putative_enhancers`	epiRomics class database containing putative enhancer calls
`database`	epiRomics class database containing all data initially loaded

Value

Variable of class epiRomics with enhanceosome annotations

Examples

```
db <- make_example_database()
eso <- make_example_enhanceosome(db)
length(annotations(eso))
```

`find_enhancers_by_comarks` *Identifies putative enhancer regions utilizing select histone mark co-occurrence*

Description

Identifies putative enhancer regions utilizing select histone mark co-occurrence

Usage

```
find_enhancers_by_comarks(
  database,
  histone_mark_1 = "h3k4me1",
  histone_mark_2 = "h3k27ac"
)
```

Arguments

database	epiRomics class database containing all data initially loaded
histone_mark_1	name of first histone mark, must match a name entry in meta(database), default set to h3k4me1
histone_mark_2	name of second histone mark, must match a name entry in meta(database), default set to h3k27ac

Value

Variable of class epiRomics further exploring candidate enhancer regions identified after histone integration

Examples

```
db <- make_example_database()
enhancers <- find_enhancers_by_comarks(db)
length(annotations(enhancers))
```

---

find_putative_enhancers

Identify putative enhancer regions using rule-based histone logic

---

Description

Automatically scans all histone and histone variant marks in the epiRomics database and applies ChromHMM-based classification rules to identify putative enhancer regions.

Usage

```
find_putative_enhancers(
  database,
  chromatin_states = NULL,
  hic_contacts = NULL,
  enhancer_states = base::c("active", "poised", "primed")
)
```

Arguments

database	An epiRomics S4 database object.
chromatin_states	data.frame or NULL. Pre-computed output from classify_chromatin_states. If NULL (default), computed automatically from all available histone marks.
hic_contacts	data.frame or NULL. Hi-C contacts in BEDPE format. Anchors are added as putative enhancers and classified using available histone data.
enhancer_states	Character vector. Chromatin states to include as putative enhancers. Default includes all enhancer-related states.

Details

This function uses a multi-source approach:

Chromatin states Leverages `classify_chromatin_states` to classify all genomic regions covered by histone marks. Regions classified as enhancer-related states are included.

Hi-C contacts If provided, Hi-C contact anchors are added as putative enhancers. Anchors are classified using the available histone data at each anchor; anchors with no histone coverage are labeled `"Unmarked"`.

TF binding Regions bound by at least one TF (type = "chip") are included as putative enhancers. TF binding alone yields `"Unmarked"` chromatin state.

H2A.Z regions H2A.Z-positive regions that were classified as `"unmarked"` by chromatin states are recovered as putative enhancers, since H2A.Z is enriched at regulatory elements (Giaimo et al. 2019; Lai & Pugh 2017). H2A.Z alone is insufficient for specific chromatin state assignment.

Unlike earlier versions that required the user to specify exactly two histone marks, this function automatically uses ALL histone marks in the database and applies the full set of classification rules.

Value

A data.frame with columns:

putative_id Integer. Unique enhancer index (sorted by TF co-binding then histone marks).

chr Character. Chromosome.

start Integer. Start position.

end Integer. End position.

width Integer. Region width.

source Character. Origin: `"histone"`, `"hic"`, `"tf"`, or comma-separated if multiple sources contribute.

chromatin_state Character. Broad state category: Active, Poised, Repressed, or Unmarked.

chromatin_state_detail Character. Specific state from `classify_chromatin_states` (e.g. active_enhancer, poised_enhancer, primed_enhancer).

histone_marks Character. Comma-separated histone marks overlapping this region.

n_histone_marks Integer. Number of histone marks.

h2az Logical. Whether H2A.Z overlaps this region.

tf_names Character. Comma-separated TF names with peaks overlapping this region (H2A.Z excluded from TF count).

n_tfs Integer. Number of TFs with binding peaks.

Examples

```
db <- make_example_database()
pe <- find_putative_enhancers(db)
head(pe[, c("chr", "start", "end", "chromatin_state")])
```

genome *Access the genome assembly name*

Description

Returns the character string stored in the genome slot (e.g. "hg38", "mm10"). This method extends the genome() generic from the **GenomeInfoDb** package.

Usage

```
## S4 method for signature 'epiRomicsS4'
genome(x)
```

Arguments

x	An epiRomicsS4 object.

Value

A character scalar. Empty character if unset.

Examples

```
db <- make_example_database()
genome(db)
```

genome-set *Replace the genome assembly name*

Description

Assigns a new genome assembly name (e.g. "hg38", "mm10") to the genome slot. This method extends the genome<-() replacement generic from the **GenomeInfoDb** package.

Usage

```
## S4 replacement method for signature 'epiRomicsS4'
genome(x) <- value
```

Arguments

x	An epiRomicsS4 object.
value	A non-empty character scalar.

Value

The updated epiRomicsS4 object.

Examples

```
db <- make_example_database()
genome(db) <- "hg38"
genome(db)
```

get_cache_path *Get Path to Cached epiRomics Data (No Download)*

Description

Returns the path to previously cached example data, or NULL if the data has not been downloaded yet.

Usage

```
get_cache_path()
```

Value

Character string path, or NULL if data is not cached.

See Also

cache_data, has_cache

Examples

```
cache_path <- get_cache_path()
if (!is.null(cache_path)) {
  list.files(cache_path)
}
```

get_regions_of_interest *Define regions of interest and filter enhanceosome by overlap*

Description

Evaluates whether regions of interest derived from external experiments (ATAC-seq, DBA, gene lists, BED files) correspond with enhanceosome regions. Supports multiple input types for flexible region definition.

Usage

```
get_regions_of_interest(
  putative_enhanceosome,
  test_regions = NULL,
  input_type = c("granges", "bed", "genelist", "combined"),
  bed_path = NULL,
  gene_list = NULL
)
```

Arguments

putative_enhanceosome	epiRomics class database containing putative enhanceosome calls. Must have non-empty annotations().
test_regions	GRanges or NULL. Direct GRanges regions of interest (original interface). If provided, input_type is ignored and this is used directly for overlap. Default NULL.
input_type	Character. How to construct test regions when test_regions is NULL. One of: "granges" Use test_regions directly (default, backward-compatible). "bed" Import regions from a BED file via bed_path. "genelist" Select enhanceosome regions whose SYMBOL column matches genes in gene_list. "combined" Union of all provided evidence sources (GRanges, BED, genelist).
bed_path	Character or NULL. Path to a BED file for input_type = "bed" or "combined".
gene_list	Character vector or NULL. Gene symbols for input_type = "genelist" or "combined".

Value

Variable of class epiRomics with enhanceosome regions overlapping with regions of interest.

Examples

```
db  <- make_example_database()
eso <- make_example_enhanceosome(db)
test_gr <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(start = 1000L, end = 60000L)
)
roi <- get_regions_of_interest(eso, test_regions = test_gr)
length(annotations(roi))
```

---

has_cache *Check Whether epiRomics Example Data is Cached*

Description

Tests whether the example data archive has been previously downloaded and extracted using cache_data. This function never triggers a download.

Usage

```
has_cache()
```

Value

Logical; TRUE if the data is cached and extracted, FALSE otherwise.

See Also

cache_data to download the data.

Examples

```
## Check data availability (does NOT download)
if (has_cache()) {
  message("Example data is available locally.")
} else {
  message("Run cache_data() to download example data.")
}
```

`make_example_bigwig` *Create a temporary synthetic BigWig file for use in examples*

Description

Writes a BigWig file from synthetic `GRanges` and numeric scores. The file is created in `base::tempdir()` by default.

Usage

```
make_example_bigwig(regions_gr = NULL, scores = NULL, path = NULL)
```

Arguments

`regions_gr`	A `GRanges` object. When `NULL` (default), a small five-region GRanges on chr1 is used.
`scores`	A numeric vector the same length as `regions_gr`. When `NULL` (default), scores 1–5 are used.
`path`	Character file path for output. When `NULL` (default), a path is generated via `base::tempfile(fileext = ".bw")`.

Details

Caller responsibility: The returned path points to a temporary file that persists for the R session. Call `file.remove(path)` when the file is no longer needed to avoid accumulating temporary files.

Value

Character string giving the path to the created BigWig file.

See Also

`make_example_database`, `rtracklayer::export`

Other synthetic example data helpers: `make_example_database()`, `make_example_enhanceosome()`, `make_example_putative_enhancers()`

Examples

```
bw_path <- make_example_bigwig()
file.exists(bw_path)
file.remove(bw_path)
```

`make_example_database` *Build a synthetic epiRomicsS4 database for use in examples*

Description

Constructs a fully populated `epiRomicsS4` object from in-memory `GRanges`, with no network access and no external files required. The returned object contains five histone marks (h3k4me1, h3k27ac, h3k27me3, h3k4me3, h3k36me3), two ChIP-seq TF tracks (TF1, TF2), and one functional annotation track (fantom), all anchored on chr1 of hg38.

Usage

```
make_example_database(genome = "hg38")
```

Arguments

`genome`	Character string naming the genome assembly (default `"hg38"`). Only the label is changed; the synthetic GRanges are always on chr1 regardless of this value, which is sufficient for example purposes.

Details

This function is the canonical data source for all `@examples` blocks in the epiRomics package. It is also used in vignettes and testthat helpers.

Value

An `epiRomicsS4` object with:

annotations GRanges containing all synthetic peak calls

meta data.frame with columns `name` and `type`

genome the value of `genome`

txdb `"TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene"`

organism `"org.Hs.eg.db"`

See Also

`make_example_putative_enhancers`, `make_example_enhanceosome`, `make_example_bigwig`

Other synthetic example data helpers: `make_example_bigwig()`, `make_example_enhanceosome()`, `make_example_putative_enhancers()`

Examples

```
db <- make_example_database()
genome(db)
nrow(meta(db))
```

make_example_enhanceosome *Build a synthetic enhanceosome result for use in examples*

Description

Returns an epiRomicsS4 object whose annotations slot holds a synthetic enhanceosome GRanges with the same column structure as the output of find_enhanceosomes (one integer count column per ChIP TF in the meta() table of database, plus a ChIP_Hits total column).

Usage

```
make_example_enhanceosome(database = NULL)
```

Arguments

database	An epiRomicsS4 object, or NULL (default). When NULL, make_example_database is called internally.

Details

The result is built directly from in-memory structures (no ChIPseeker annotation, no TxDb lookup) so the example completes in under one second under R CMD check. It is fit-for-purpose for exercising the enhanceosome-consuming functions (analyze_tf_cobinding, analyze_tf_overlap, etc.) without the cost of the full enhancer-calling pipeline.

Value

An epiRomicsS4 object whose annotations slot contains the synthetic enhanceosome GRanges.

See Also

make_example_database, find_enhanceosomes

Other synthetic example data helpers: make_example_bigwig(), make_example_database(), make_example_putative_enhancers()

Examples

```
eso <- make_example_enhanceosome()
length(annotations(eso))
```

make_example_putative_enhancers *Build a synthetic putative-enhancer result for use in examples*

Description

Returns a `data.frame` matching the output format of `find_putative_enhancers`. When `database` is `NULL` (the default), a fresh synthetic database is created internally via `make_example_database` and then `find_putative_enhancers` is called on it to produce a genuine result. This keeps the helper fast (< 2 s) while ensuring the output structure is always consistent with the real function.

Usage

```
make_example_putative_enhancers(database = NULL)
```

Arguments

`database`	An `epiRomicsS4` object, or `NULL` (default). When `NULL`, a synthetic database is built automatically.

Value

A `data.frame` with columns produced by `find_putative_enhancers`: `putative_id`, `chr`, `start`, `end`, `width`, `source`, `chromatin_state`, `chromatin_state_detail`, `histone_marks`, `n_histone_marks`, `h2az`, `tf_names`, `n_tfs`.

See Also

`make_example_database`, `find_putative_enhancers`

Other synthetic example data helpers: `make_example_bigwig()`, `make_example_database()`, `make_example_enhanceosome()`

Examples

```
pe <- make_example_putative_enhancers()
nrow(pe)
head(pe[, c("chr", "start", "end", "chromatin_state")])
```

maxCovBwCached *BigWig coverage calculation*

Description

Computes maximum coverage from a BigWig file over specified genomic regions using region-specific BigWigSelection import. Uses `na.rm = TRUE` for robustness with potentially missing data.

Usage

```
maxCovBwCached(bw_path, gr)
```

Arguments

bw_path	Character string path to BigWig file
gr	GenomicRanges object containing regions

Value

Numeric value representing the maximum coverage across the regions

Examples

```
bw_path <- make_example_bigwig()
gr <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(1000L, 2000L)
)
max_cov <- maxCovBwCached(bw_path, gr)
file.remove(bw_path)
```

---

maxCovFilesCached *Multiple BigWig coverage calculation*

Description

Multi-file BigWig coverage calculation with parallel processing and batch query support. Uses region-specific BigWig import.

Usage

```
maxCovFilesCached(bw_paths, gr, parallel = FALSE, fast = FALSE)
```

Arguments

bw_paths	Character vector of paths to BigWig files
gr	GenomicRanges object containing regions
parallel	Logical, whether to use parallel processing (default: FALSE)
fast	Logical, whether to use batch BigWig R-tree query path for all regions at once per file (default: FALSE). When TRUE, uses `.fast_bw_signal()` for efficient multi-region queries.

Value

GenomicRanges object with coverage values added as metadata column X

Examples

```
bw_path <- make_example_bigwig()
gr <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(1000L, 2000L)
)
result <- maxCovFilesCached(c(bw_path, bw_path), gr)
file.remove(bw_path)
```

meta — *Access the metadata slot of an epiRomicsS4 object*

Description

Returns the `data.frame` stored in the `meta` slot, listing the name and type of every data source loaded into the database. Public accessor replacement for `obj@meta` or `methods::slot(obj, "meta")`.

Usage

```
meta(x, ...)

## S4 method for signature 'epiRomicsS4'
meta(x, ...)
```

Arguments

`x`	An `epiRomicsS4` object.
`...`	Currently unused; reserved for method extension.

Value

A `data.frame` with one row per data source. Empty (zero rows) if no metadata has been assigned.

See Also

`meta<-`, `epiRomicsS4`

Examples

```
db <- make_example_database()
meta(db)
nrow(meta(db))
```

meta-set — *Replace the metadata slot of an epiRomicsS4 object*

Description

Assigns a new `data.frame` to the `meta` slot.

Usage

```
meta(x, ...) <- value

## S4 replacement method for signature 'epiRomicsS4'
meta(x, ...) <- value
```

Arguments

x	An epiRomicsS4 object.
...	Currently unused; reserved for method extension.
value	A data.frame.

Value

The updated epiRomicsS4 object.

Examples

```
db <- make_example_database()
new_meta <- meta(db)
new_meta$extra <- "annotated"
meta(db) <- new_meta
colnames(meta(db))
```

---

organism *Access the organism annotation package name*

Description

Returns the character string stored in the organism slot (e.g. "org.Hs.eg.db", "org.Mm.eg.db"). This method extends the organism() generic from the **BiocGenerics** package.

Usage

```
## S4 method for signature 'epiRomicsS4'
organism(object)
```

Arguments

object	An epiRomicsS4 object.

Value

A character scalar. Empty character if unset.

Examples

```
db <- make_example_database()
organism(db)
```

organism-set *Replace the organism annotation package name*

Description

Assigns a new `org.*.eg.db` package name to the `organism` slot. This method extends the `organism<-()` replacement generic from the **BiocGenerics** package.

Usage

```
## S4 replacement method for signature 'epiRomicsS4'
organism(object) <- value
```

Arguments

`object`	An `epiRomicsS4` object.
`value`	A character scalar, typically an `org.*.eg.db` package name.

Value

The updated `epiRomicsS4` object.

Examples

```
db <- make_example_database()
organism(db) <- "org.Hs.eg.db"
organism(db)
```

plot_gene_tracks *Visualize any gene locus with multi-track BigWig overlay*

Description

A convenience wrapper around `plot_tracks` that looks up a gene by symbol and creates a multi-track view without requiring the full enhanceosome pipeline.

Usage

```
plot_gene_tracks(
  gene_symbol,
  database,
  track_connection,
  chromatin_states = NULL,
  padding = 1000L,
  show_bigwig = TRUE,
  show_chromatin = TRUE,
  show_annotations = TRUE,
  show_gene_model = TRUE,
  show_enhancer_highlight = TRUE,
  mirror = TRUE,
```

```
  signal_style = c("line", "polygon"),
  signal_layout = "auto",
  cex_cell_label = 1.4,
  cex_axis = 1.2,
  cex_coord = 1.3,
  cex_annotation = 1.1,
  cex_gene = 1.2,
  cex_title = 1.5,
  cex_signal_label = 1.2,
  quantile_cap = 0.98,
  scale_factor = 1.1,
  export = NULL,
  width = 10,
  height = 8
)
```

Arguments

gene_symbol	Character. HGNC gene symbol (e.g. "INS", "GCG", "PDX1").
database	An epiRomics S4 database object.
track_connection	A data.frame from the BigWig CSV sheet (columns: path, name, color, type).
chromatin_states	Optional. Output of classify_chromatin_states.
padding	Integer. Base pairs of padding around the gene body (default 1000).
show_bigwig	Logical. Show BigWig signal tracks (default TRUE).
show_chromatin	Logical. Show chromatin state tracks (default TRUE).
show_annotations	Logical. Show BED annotation tracks (default TRUE).
show_gene_model	Logical. Show gene model panel (default TRUE).
show_enhancer_highlight	Logical. Show enhancer highlight (default TRUE).
mirror	Logical. Enable mirrored ATAC/RNA layout (default TRUE).
signal_style	Character. Signal rendering style: "line" (default) draws vertical bars at each position (IGV/UCSC browser style), "polygon" draws filled area charts.
signal_layout	Character. Signal layout mode: "auto", "stacked", or "mirrored".
cex_cell_label	Numeric. Font size for cell type labels (default 1.4).
cex_axis	Numeric. Font size for axis labels (default 1.2).
cex_coord	Numeric. Font size for coordinate bar (default 1.3).
cex_annotation	Numeric. Font size for annotation labels (default 1.1).
cex_gene	Numeric. Font size for gene model labels (default 1.2).
cex_title	Numeric. Font size for plot title (default 1.5).
cex_signal_label	Numeric. Font size for signal indicators (default 1.2).
quantile_cap	numeric. Percentile for capping extreme signal peaks (default 0.98). Peaks above this percentile are clipped to prevent axis compression.

scale_factor	numeric. Y-axis headroom multiplier (default 1.1). Values above 1.0 add whitespace above the tallest signal peak.
export	Character or NULL. File path for export (pdf, eps, png, tiff).
width	Numeric. Export width in inches (default 10).
height	Numeric. Export height in inches (default 8).

Details

The function queries the TxDb for the official gene body coordinates, builds a synthetic single-region epiRomics object, and passes it to `plot_tracks` for rendering. The viewing window spans the gene body plus `padding` on each side.

Value

Invisible NULL. A multi-panel base-R plot is drawn on the current graphics device.

Examples

```
## plot_gene_tracks requires a TxDb (Suggests) and a real BigWig file.
## This example confirms input validation fires correctly.
db <- make_example_database()
tc <- data.frame(path = character(), name = character(),
  color = character(), type = character(),
  stringsAsFactors = FALSE)
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
  tryCatch(
    plot_gene_tracks("INS", db, tc),
    error = function(e) message(e$message)
  )
}
```

---

plot_quick_view *Quick standalone gene/region visualization from BigWig files*

Description

A zero-configuration entry point for visualizing any genomic locus with BigWig signal tracks. Requires only a gene symbol (or genomic coordinates) and one or more BigWig file paths. No database setup, no track connection CSV, no chromatin states — just signal overlaid on the gene model.

Usage

```
plot_quick_view(
  gene = NULL,
  region = NULL,
  bw_paths,
  labels = NULL,
  colors = NULL,
  genome = "hg38",
  txdb = NULL,
  organism = NULL,
  padding = 5000L,
```

```
  mirror = FALSE,
  signal_style = c("line", "polygon"),
  signal_layout = "auto",
  cex_cell_label = 1.4,
  cex_axis = 1.2,
  cex_coord = 1.3,
  cex_annotation = 1.1,
  cex_gene = 1.2,
  cex_title = 1.5,
  cex_signal_label = 1.2,
  quantile_cap = 0.98,
  scale_factor = 1.1,
  export = NULL,
  width = 10,
  height = 8
)
```

Arguments

gene	Character or NULL. HGNC gene symbol (e.g., "INS", "GCG"). Exactly one of gene or region must be provided.
region	List or NULL. Genomic coordinates as list(chr = "chr11", start = 2159779,end = 2161209). Exactly one of gene or region must be provided.
bw_paths	Named character vector of BigWig file paths. Names are used as sample labels. If names contain both "atac" and "rna" (case-insensitive), mirror pairing is attempted.
labels	Character vector or NULL. Override sample labels. If NULL, uses names(bw_paths).
colors	Character vector or NULL. Override sample colors. If NULL, uses a default colorblind-friendly palette. Must have the same length as bw_paths.
genome	Character. String name of the genome assembly associated with bw_paths (e.g. "hg38", "mm10", "rn6", "dm6"). The value must match the assembly recorded in the supplied txdb; a validator compares the string against GenomeInfoDb::genome(txdb). Defaults to "hg38" for convenience with built-in fallbacks.
txdb	Character or NULL. TxDb specification in "package::object" form (e.g. "TxDb.Rnorvegicus.UCS TxDb.Rnorvegicus.UCSC.rn6.refGene"). If NULL, auto-resolved for the built-in convenience genomes "hg38" and "mm10"; for any other genome, txdb must be supplied explicitly.
organism	Character or NULL. org.db specification (e.g. "org.Rn.eg.db"). If NULL, auto-resolved for the built-in convenience genomes "hg38" and "mm10"; for any other genome, organism must be supplied explicitly.
padding	Integer. Base pairs of padding around gene/region (default 5000).
mirror	Logical. Enable mirrored ATAC/RNA layout (default FALSE).
signal_style	Character. Signal rendering style: "line" (default) draws vertical bars at each position (IGV/UCSC browser style), "polygon" draws filled area charts.
signal_layout	Character. Signal layout mode: "auto", "stacked", or "mirrored".
cex_cell_label	Numeric. Font size for cell type labels (default 1.4).
cex_axis	Numeric. Font size for axis labels (default 1.2).
cex_coord	Numeric. Font size for coordinate bar (default 1.3).

cex_annotation	Numeric. Font size for annotation labels (default 1.1).
cex_gene	Numeric. Font size for gene model labels (default 1.2).
cex_title	Numeric. Font size for plot title (default 1.5).
cex_signal_label	Numeric. Font size for signal indicators (default 1.2).
quantile_cap	numeric. Percentile for capping extreme signal peaks (default 0.98). Peaks above this percentile are clipped to prevent axis compression.
scale_factor	numeric. Y-axis headroom multiplier (default 1.1). Values above 1.0 add whitespace above the tallest signal peak.
export	Character or NULL. File path for export (pdf, eps, png, tiff).
width	Numeric. Export width in inches (default 10).
height	Numeric. Export height in inches (default 8).

Details

This is the simplest way to use epiRomics for exploratory visualization. Specify a gene by symbol (e.g., "INS") or a region by coordinates. The function auto-resolves the TxDb annotation database for the specified genome assembly.

Value

Invisible NULL. A multi-panel base-R plot is drawn on the current graphics device (or exported to file if export is set).

Examples

```
## plot_quick_view requires a TxDb (Suggests) and a real BigWig file.
## This example confirms that input validation fires correctly.
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
  bw_path <- make_example_bigwig()
  tryCatch(
    plot_quick_view(
      region = list(chr = "chr1", start = 1000L, end = 51000L),
      bw_paths = c(Sample = bw_path),
      genome   = "hg38"
    ),
    error = function(e) message(e$message)
  )
  file.remove(bw_path)
}
```

---

plot_signal_histogram *Plot signal distribution histogram for accessibility threshold selection*

Description

Visualizes the distribution of BigWig signal across genomic regions to help identify the noise floor and select an appropriate z-score or raw signal threshold for calling accessible regions. The distribution is typically bimodal: a large peak at zero/low signal (background noise) and a smaller peak at higher signal (accessible regions).

Usage

```
plot_signal_histogram(
  bw_paths,
  regions,
  n_bins = 100,
  show_z_lines = base::c(1, 1.5, 2),
  log_scale = TRUE,
  cex_label = 0.7,
  cex_title = 1,
  hist_fill = "#CCDDEE",
  hist_border = "#88AACC",
  density_col = "#2C3E50",
  z_colors = base::c("#2ECC40", "#FF851B", "#E74C3C"),
  suggest_col = "#B10DC9"
)
```

Arguments

bw_paths	named character vector of BigWig file paths.
regions	GRanges object of regions to examine.
n_bins	integer, number of histogram bins (default: 100).
show_z_lines	numeric vector, z-score thresholds to show as vertical lines (default: c(1.0, 1.5, 2.0)).
log_scale	logical, use log10(signal + 1) for x-axis (default: TRUE).
cex_label	Numeric. Font size for threshold and annotation labels (default 0.7).
cex_title	Numeric. Font size for histogram titles (default 1.0).
hist_fill	character. Histogram bar fill color (default "#CCDDEE").
hist_border	character. Histogram bar border color (default "#88AACC").
density_col	character. Density overlay line color (default "#2C3E50").
z_colors	character vector. Colors for z-score threshold lines (default: green, orange, red).
suggest_col	character. Color for suggested threshold line (default "#B10DC9").

Value

Invisible list of per-sample signal statistics, including mean, sd, quantiles, and suggested thresholds.

Examples

```
bw_path <- make_example_bigwig()
regions <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(
    start = c(1000L, 5000L, 10000L, 20000L, 50000L),
    end   = c(2000L, 6000L, 11000L, 21000L, 51000L)
  )
)
stats <- plot_signal_histogram(c(Sample = bw_path), regions)
file.remove(bw_path)
```

plot_tracks *Multi-track genomic visualization (base R graphics)*

Description

Renders a stacked multi-panel genome browser view using base R graphics. Includes gene model, BigWig signal tracks (ATAC, RNA, histone), enhancer index, chromatin state bars, FANTOM/UCNE annotations, TF binding peaks, and ncRNA annotations. A translucent violet highlight column marks the enhancer region of interest across all panels. Typically renders in 1-2 seconds.

Usage

```
plot_tracks(
  putative_enhanceosome,
  index,
  database,
  track_connection,
  keep_epitracks = TRUE,
  chromatin_states = NULL,
  gene_lookup = FALSE,
  show_bigwig = TRUE,
  show_chromatin = TRUE,
  show_annotations = TRUE,
  show_gene_model = TRUE,
  show_enhancer_highlight = TRUE,
  mirror = TRUE,
  signal_style = c("line", "polygon"),
  signal_layout = "auto",
  cex_cell_label = 1.4,
  cex_axis = 1.2,
  cex_coord = 1.3,
  cex_annotation = 1.1,
  cex_gene = 1.2,
  cex_title = 1.5,
  cex_signal_label = 1.2,
  quantile_cap = 0.98,
  scale_factor = 1.1,
  export = NULL,
  width = 10,
  height = 8
)

plot_tracks_fast(
  putative_enhanceosome,
  index,
  database,
  track_connection,
  keep_epitracks = TRUE,
  chromatin_states = NULL,
  gene_lookup = FALSE,
  show_bigwig = TRUE,
```

```
  show_chromatin = TRUE,
  show_annotations = TRUE,
  show_gene_model = TRUE,
  show_enhancer_highlight = TRUE,
  mirror = TRUE,
  signal_style = c("line", "polygon"),
  signal_layout = "auto",
  cex_cell_label = 1.4,
  cex_axis = 1.2,
  cex_coord = 1.3,
  cex_annotation = 1.1,
  cex_gene = 1.2,
  cex_title = 1.5,
  cex_signal_label = 1.2,
  quantile_cap = 0.98,
  scale_factor = 1.1,
  export = NULL,
  width = 10,
  height = 8
)
```

Arguments

`putative_enhanceosome`	epiRomics class database containing putative enhanceosome calls
`index`	numeric of row value from putative_enhanceosome to visualize
`database`	epiRomics class database containing all data initially loaded
`track_connection`	data frame containing bigwig track locations and their names
`keep_epitracks`	logical indicating whether to show enhancer and chip tracks, default is TRUE
`chromatin_states`	data.frame, optional output from `classify_chromatin_states`. When provided, the enhancer track is colored by chromatin state and separate per-state annotation tracks are added.
`gene_lookup`	logical. When TRUE, omits the enhancer index bar and violet highlight column. Used internally by `plot_gene_tracks` to display a gene locus without enhancer-specific visual elements. Default is FALSE.
`show_bigwig`	logical. Show BigWig signal tracks. Default TRUE.
`show_chromatin`	logical. Show chromatin state color tracks. Default TRUE.
`show_annotations`	logical. Show BED annotation tracks. Default TRUE.
`show_gene_model`	logical. Show gene model panel. Default TRUE.
`show_enhancer_highlight`	logical. Show enhancer index highlight. Default TRUE.
`mirror`	logical. Use symmetric mirrored axes for paired ATAC/RNA signals. Default TRUE.
`signal_style`	character. Signal rendering style: `"line"` (default) draws vertical bars at each position (IGV/UCSC browser style), `"polygon"` draws filled area charts.

signal_layout	character. Signal rendering layout: "auto" detects paired signals and mirrors, "stacked" renders one row per signal, "mirrored" always mirrors first two. Default "auto".
cex_cell_label	numeric. Font size for cell type labels (e.g. Alpha, Beta). Default 1.4.
cex_axis	numeric. Font size for y-axis tick labels on signal tracks. Default 1.2.
cex_coord	numeric. Font size for chromosome, start, stop, genome build text. Default 1.3.
cex_annotation	numeric. Font size for BED annotation labels. Default 1.1.
cex_gene	numeric. Font size for gene name labels in gene model. Default 1.2.
cex_title	numeric. Font size for main plot title. Default 1.5.
cex_signal_label	numeric. Font size for ATAC/RNA type indicator text. Default 1.2.
quantile_cap	numeric. Percentile for capping extreme signal peaks (default 0.98). Peaks above this percentile are clipped to prevent axis compression.
scale_factor	numeric. Y-axis headroom multiplier (default 1.1). Values above 1.0 add whitespace above the tallest signal peak.
export	character or NULL. File path to auto-export the plot (e.g. "plot.pdf", "plot.eps", "plot.png"). When NULL (default), plot is drawn on current device only.
width	numeric. Export width in inches. Default 10.
height	numeric. Export height in inches. Default 8.

Value

Invisible NULL. The function produces a base R multi-panel plot on the current graphics device.

Examples

```
## plot_tracks requires a TxDb (Suggests) and a real BigWig file.
## This example confirms input validation fires correctly.
db <- make_example_database()
eso <- make_example_enhanceosome(db)
tc  <- data.frame(path = character(), name = character(),
  color = character(), type = character(),
  stringsAsFactors = FALSE)
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
  tryCatch(
    plot_tracks(eso, 1L, db, tc),
    error = function(e) message(e$message)
  )
}
```

---

txdb *Access the TxDb package::object identifier slot*

Description

Returns the character string stored in the txdb slot. The value has the form "TxDb.Package::TxDb.object", e.g. "TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene". Public accessor replacement for obj@txdb or methods::slot(obj, "txdb").

Usage

```
txdb(x, ...)

## S4 method for signature 'epiRomicsS4'
txdb(x, ...)
```

Arguments

x	An epiRomicsS4 object.
...	Currently unused; reserved for method extension.

Value

A character scalar. Empty character if unset.

See Also

txdb<-, epiRomicsS4

Examples

```
db <- make_example_database()
txdb(db)
```

---

txdb-set *Replace the TxDb identifier slot of an epiRomicsS4 object*

Description

Assigns a new "TxDb.Package::TxDb.object" character string to the txdb slot. Validity is checked via validObject.

Usage

```
txdb(x, ...) <- value

## S4 replacement method for signature 'epiRomicsS4'
txdb(x, ...) <- value
```

Arguments

x	An epiRomicsS4 object.
...	Currently unused; reserved for method extension.
value	A character scalar in "TxDb.Package::TxDb.object" form.

Value

The updated epiRomicsS4 object.

Examples

```
db <- make_example_database()
txdb(db) <- "TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene"
txdb(db)
```

Index