Package 'gVenn'

December 3, 2025

Title Proportional Venn and UpSet Diagrams for Gene Sets and Genomic Regions

Version 1.1.0

Description Tools to compute and visualize overlaps between gene sets or genomic regions. Venn diagrams with proportional areas are provided, while UpSet plots are recommended for larger numbers of sets. The package supports GRanges and GRangesList inputs, and integrates with analysis workflows for ChIP-seq, ATAC-seq, and other genomic interval data. It generates clean, interpretable, and publication-ready figures.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/ckntav/gVenn, https://ckntav.github.io/gVenn/

BugReports https://github.com/ckntav/gVenn/issues

Suggests testthat (>= 3.0.0), ggplot2, withr, knitr, rmarkdown

Config/testthat/edition 3

Imports ComplexHeatmap, eulerr, GenomicRanges, IRanges, lubridate, methods, rtracklayer, stringr, writexl

biocViews Software, Visualization, ChIPSeq, ATACSeq, Epigenetics, DataRepresentation, Sequencing

VignetteBuilder knitr

Depends R (>= 4.5.0)

LazyData false

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

git_branch devel

git_last_commit 617952c

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-02

Author Christophe Tav [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8808-9617>)

Maintainer Christophe Tav <christophe.tav@gmail.com>

Contents

`a549_chipseq_peaks` *A549 ChIP-seq Consensus Peak Subsets (Dex, chr7)*

Description

Example consensus peak subsets for MED1, BRD4, and GR after dexamethasone treatment in A549 cells. Each set has been restricted to peaks on `chr7` to keep the dataset small and suitable for examples and tests.

Usage

```
a549_chipseq_peaks
```

Format

A `GRangesList` with 3 named elements:

MED1_Dex_chr7 Consensus MED1 peaks (chr7 subset).

BRD4_Dex_chr7 Consensus BRD4 peaks (chr7 subset).

GR_Dex_chr7 Consensus GR peaks (chr7 subset).

Details

The original full consensus peak sets are available as gzipped BED files in `inst/extdata/`:

- `A549_MED1_Dex.stdchr.bed.gz`
- `A549_BRD4_Dex.stdchr.bed.gz`
- `A549_GR_Dex.stdchr.bed.gz`

These are not trimmed, but for package efficiency the dataset here (`a549_chipseq_peaks`) only includes the chr7 subsets.

Source

Internal consensus peak sets processed in A549 cells after dexamethasone stimulation.

References

Tav C, Fournier É, Fournier M, Khadangi F, Baguette A, Côté MC, Silveira MAD, Bérubé-Simard F-A, Bourque G, Droit A, Bilodeau S (2023). "Glucocorticoid stimulation induces regionalized gene responses within topologically associating domains." *Frontiers in Genetics*. doi:10.3389/fgene.2023.1237092

Examples

```
# Load dataset
data(a549_chipseq_peaks)
a549_chipseq_peaks

# Compute overlaps and plot
ov <- computeOverlaps(a549_chipseq_peaks)
plotVenn(ov)
```

`computeOverlaps` *Compute Overlaps Between Multiple Sets or Genomic Regions*

Description

`computeOverlaps()` is the main entry point for overlap analysis. It accepts either genomic region objects (`GRanges`/`GRangesList`) or ordinary sets (character/numeric vectors) and computes a binary overlap matrix describing the presence or absence of each element across sets.

Usage

```
computeOverlaps(x)
```

Arguments

x	Input sets. One of: • A `GRangesList` object. • A named list of `GRanges` objects. • A named list of atomic vectors (character, numeric, factor, etc.), all of the same type.

Details

- When provided with genomic regions, the function merges all intervals into a non-redundant set (`reduce()`), then determines which original sets each region overlaps.
- When provided with ordinary sets (e.g., gene symbols), it collects all unique elements and records which sets contain them.

The resulting object encodes both the overlap matrix and compact category labels (e.g., `"110"`) representing the overlap pattern of each element. These results can be directly passed to visualization functions such as `plotVenn()` or `plotUpSet()`.

Internally, `computeOverlaps()` dispatches to either `computeGenomicOverlaps()` (for genomic inputs) or `computeSetOverlaps()` (for ordinary sets). Users are encouraged to call only `computeOverlaps()`.

Value

An S3 object encoding the overlap result whose class depends on the input type:

GenomicOverlapResult Returned when the input is genomic (`GRangesList` or list of `GRanges`). A list with:

- `reduced_regions`: A `GRanges` object containing the merged (non-redundant) intervals. Each region is annotated with an `intersect_category` column.
- `overlap_matrix`: A logical matrix indicating whether each reduced region overlaps each input set (rows = regions, columns = sets).

SetOverlapResult Returned when the input is a list of atomic vectors. A list with:

- `unique_elements`: Character vector of all unique elements across the sets.
- `overlap_matrix`: A logical matrix indicating whether each element is present in each set (rows = elements, columns = sets).
- `intersect_category`: Character vector of category codes (e.g., `"110"`) for each element.

See Also

`plotVenn`, `plotUpSet`, `GRangesList`, `reduce`

Examples

```
# Example with gene sets (built-in dataset)
data(gene_list)
ov_sets <- computeOverlaps(gene_list)
head(ov_sets$overlap_matrix)
```

```
plotVenn(ov_sets)

# Example with genomic regions (built-in dataset)
data(a549_chipseq_peaks)
ov_gr <- computeOverlaps(a549_chipseq_peaks)
head(ov_gr$overlap_matrix)
plotVenn(ov_gr)
```

`exportOverlaps` *Export Overlap Groups to Excel*

Description

This function exports the output of `extractOverlaps()` to an Excel file, creating one sheet per overlap group. Genomic overlaps (`GRanges`) are converted to data frames before export.

Usage

```
exportOverlaps(
  grouped,
  output_dir = ".",
  output_file = "overlap_groups",
  with_date = TRUE,
  verbose = TRUE
)
```

Arguments

`grouped`	Overlap groups from `extractOverlaps()`.
`output_dir`	A string specifying the output directory. Defaults to `"."`.
`output_file`	A string specifying the base filename (without extension). Defaults to `"overlap_groups"`.
`with_date`	Logical (default `TRUE`). Whether to prepend the current date (from `today`) to the filename.
`verbose`	Logical. If `TRUE`, print a message with the saved path. Default `TRUE`.

Value

Overlap groups are saved to a Excel file on disk. Invisibly returns the full path to the saved file.

Examples

```
res <- computeOverlaps(list(A = letters[1:3], B = letters[2:4]))
grouped <- extractOverlaps(res)
exportOverlaps(grouped, output_dir = tempdir(), output_file = "overlap_groups")
```

`exportOverlapsToBed` *Export Overlap Groups to BED Files*

Description

This function exports genomic overlap groups from `extractOverlaps()` to BED format files, creating one BED file per overlap group.

Usage

```
exportOverlapsToBed(
  grouped,
  output_dir = ".",
  output_prefix = "overlaps",
  with_date = TRUE,
  verbose = TRUE
)
```

Arguments

`grouped`	Genomic overlap groups from `extractOverlaps()` (must be `GRangesList`).
`output_dir`	A string specifying the output directory. Defaults to `"."`.
`output_prefix`	A string specifying the filename prefix. Defaults to `"overlaps"`.
`with_date`	Logical (default `TRUE`). Whether to prepend the current date to filenames.
`verbose`	Logical. If `TRUE`, print messages. Default `TRUE`.

Details

This function only works with genomic overlaps (i.e., when the input to `extractOverlaps()` was a `GenomicOverlapResult` object, resulting in a `GRangesList`). It does not work with set overlaps (character vectors). Each overlap group will be saved as a separate BED file with the group identifier included in the filename.

Value

Invisibly returns a character vector of file paths created.

extractOverlaps *Extract Overlap Groups from Genomic or Set Overlap Results*

Description

This function extracts subsets of intersecting elements grouped by their overlap category (e.g., "110"). For genomic overlaps, it returns a `GRangesList`; for set overlaps, it returns a named list of character vectors.

Usage

```
extractOverlaps(overlap_object)
```

Arguments

`overlap_object` A `GenomicOverlapsResult` or `SetOverlapsResult` object.

Value

A named list of grouped intersecting elements:

- If input is a `GenomicOverlapsResult`, a `GRangesList` split by `intersect_category`.
- If input is a `SetOverlapsResult`, a named `list` of character vectors grouped by `intersect_category`.

Examples

```
# Example with gene sets (built-in dataset)
data(gene_list)
res_sets <- computeOverlaps(gene_list)
group_gene <- extractOverlaps(res_sets)
group_gene

# Example with genomic regions (built-in dataset)
data(a549_chipseq_peaks)
res_genomic <- computeOverlaps(a549_chipseq_peaks)
group_genomic <- extractOverlaps(res_genomic)
group_genomic
```

gene_list *Example Gene Lists with Overlaps*

Description

A synthetic dataset of three gene lists, created from the first 250 human gene symbols in **org.Hs.eg.db**.

Usage

```
gene_list
```

Format

A named `list` of length 3. Each element is a character vector of gene symbols:

random_genes_A 125 gene symbols.

random_genes_B 115 gene symbols.

random_genes_C 70 gene symbols.

Source

Generated from **org.Hs.eg.db** (keys of type `SYMBOL`), using a reproducible random seed.

Examples

```
data(gene_list)

# Inspect the list
str(gene_list)

# Compute overlaps and plot
ov <- computeOverlaps(gene_list)
plotVenn(ov)
```

`plotUpSet` *Plot an UpSet Diagram from Genomic or Set Overlap Results*

Description

This function creates an UpSet plot using the `ComplexHeatmap` package to visualize intersections across multiple sets. Supports both `GenomicOverlapsResult` and `SetOverlapsResult` objects.

Usage

```
plotUpSet(overlap_object, customSetOrder = NULL, comb_col = "black")
```

Arguments

`overlap_object`	A `GenomicOverlapsResult` or `SetOverlapsResult` object returned by `computeOverlaps`.
`customSetOrder`	Optional. A vector specifying the order of sets to display on the UpSet diagram. The vector should contain either numeric indices (corresponding to the sets in the overlap object) or character names (matching the set names). If `NULL` (default), sets are displayed in decreasing order of their size (`set_size()`).
`comb_col`	Optional. Color(s) for the combination matrix dots and connecting lines. Can be a single color, a vector of colors (recycled to match the number of intersections). Default is "black".

Value

An UpSet plot object generated by `ComplexHeatmap::UpSet`.

Examples

```
# Example with gene sets (built-in dataset)
data(gene_list)
res_sets <- computeOverlaps(gene_list)

# Default order (sets sorted by size)
plotUpSet(res_sets)

# Custom color
plotUpSet(res_sets, comb_col = "darkblue")

# Custom order by names
plotUpSet(res_sets, customSetOrder = c("random_genes_C",
                                       "random_genes_A",
                                       "random_genes_B"))

# Example with genomic regions (built-in dataset)
data(a549_chipseq_peaks)
res_genomic <- computeOverlaps(a549_chipseq_peaks)
plotUpSet(res_genomic)
```

`plotVenn` *Plot a Venn Diagram from Genomic or Set Overlap Results*

Description

This function creates a Venn diagram using the `eulerr` package to visualize intersections across multiple sets. Supports both `GenomicOverlapsResult` and `SetOverlapsResult` objects.

Usage

```
plotVenn(
  overlap_object,
  fills = TRUE,
  edges = TRUE,
  labels = FALSE,
  quantities = list(type = "counts"),
  legend = "right",
  main = NULL,
  ...
)
```

Arguments

`overlap_object`	A `GenomicOverlapsResult` or `SetOverlapsResult` object returned by `computeOverlaps`.
`fills`	Controls the fill appearance of the diagram. Can be: • **logical**: `TRUE` (default) shows fills, `FALSE` hides them • **character vector**: Colors for the fills. Default colors are: c("#2B70AB", "#FFB027", "#3EA742", "#CD3301", "#9370DB", "#008B8B", "#D87093") • **list**: Fine control with graphical parameters including `fill` (colors), `alpha` (transparency 0-1)
`edges`	Controls the edge/border appearance. Can be: • **logical**: `TRUE` (default) shows edges, `FALSE` hides them • **character vector**: Colors for the edges • **list**: Fine control with `col` (colors), `alpha` (transparency 0-1), `lty` (line type), `lwd` (line width), `lex` (line expansion)
`labels`	Controls set labels. Can be: • **logical**: `TRUE` shows default labels, `FALSE` hides them • **character vector**: Custom text for labels • **list**: Fine control with `col` (text color), `fontsize`, `font` (1=plain, 2=bold, 3=italic, 4=bold italic), `fontfamily`, `cex` (character expansion), `alpha` (transparency 0-1)
`quantities`	Controls intersection quantities display. Can be: • **logical**: `TRUE` shows counts, `FALSE` hides them • **character vector**: Custom text labels • **list**: Fine control with `type` (c("counts", "percent")), `col` (text color), `fontsize`, `font`, `fontfamily`, `cex`, `alpha`
`legend`	Controls the legend. Can be: • **logical**: `FALSE` to disable • **character**: Position ("right", "top", "bottom", "left") • **list**: Fine control with `side` (position), `labels` (custom labels), `col`, `fontsize`, `cex`, `fontfamily`
`main`	Title for the plot. Can be character, expression, or list with `label` (text), `col`, `fontsize`, `font`, `fontfamily`
`...`	Additional arguments passed to `plot.euler`.

Value

A Venn diagram plot generated by `eulerr`.

Examples

```
# Example with gene sets
data(gene_list)
res_sets <- computeOverlaps(gene_list)
```

```
# Basic plot
plotVenn(res_sets)

# Customize fills with transparency and custom colors
plotVenn(res_sets,
         fills = list(fill = c("#FF6B6B", "#4ECDC4", "#45B7D1"),
                      alpha = 0.6))

# Customize edges
plotVenn(res_sets,
         edges = list(col = "darkgray", lwd = 2, lty = 2))

# Customize labels
plotVenn(res_sets,
         labels = list(col = "white", font = 2, fontsize = 14))

# Show both counts and percentages
plotVenn(res_sets,
         quantities = list(type = c("counts", "percent"),
                           col = "black", fontsize = 10))

# Add a title
plotVenn(res_sets,
         main = list(label = "Gene Set Overlaps",
                     col = "navy", fontsize = 16, font = 2))

# Transparent fills with colored borders only
plotVenn(res_sets,
         fills = "transparent",
         edges = list(col = c("red", "blue", "green"), lwd = 3))

# Custom legend
plotVenn(res_sets,
         legend = list(side = "bottom",
                       labels = c("Treatment A", "Treatment B", "Control"),
                       fontsize = 12))
```

`saveViz` *Save a Visualization to File (PDF, PNG, or SVG)*

Description

This function saves a visualization object to a file in the specified format and directory. It supports visualizations generated by `plotVenn()`, `plotUpSet()`, `ggplot2`, or any other plot object that can be rendered using `print()` inside a graphics device. Optionally, the current date (stored in the `today` variable) can be prepended to the filename.

Usage

```
saveViz(
```

```
  viz,
  output_dir = ".",
  output_file = "figure_gVenn",
  format = "pdf",
  with_date = TRUE,
  width = 5,
  height = 5,
  resolution = 300,
  verbose = TRUE
)
```

Arguments

viz	A visualization object typically created by either plotVenn() or plotUpSet(), but can also be a ggplot2 plot or any other plot object printable with print().
output_dir	A string specifying the output directory. Defaults to ".".
output_file	A string specifying the base filename (without extension). Defaults to "viz_genomicVenn".
format	Output format. One of "pdf", "png", or "svg". Defaults to "pdf".
with_date	Logical (default TRUE). Whether to prepend the current date (from today) to the filename.
width	Width of the output file in inches. Default is 5.
height	Height of the output file in inches. Default is 5.
resolution	Resolution in DPI (only used for PNG). Default is 300.
verbose	Logical. If TRUE, print a message with the saved path. Default TRUE.

Value

The visualization is saved to a file on disk. Invisibly returns the full path to the saved file.

Examples

```
# Example with a built-in set dataset
  data(gene_list)
  ov_sets <- computeOverlaps(gene_list)
  venn_plot <- plotVenn(ov_sets)
  saveViz(venn_plot, output_dir = tempdir(), output_file = "venn_sets")

  # Example with a built-in genomic dataset
  data(a549_chipseq_peaks)
  ov_genomic <- computeOverlaps(a549_chipseq_peaks)
  upset_plot <- plotUpSet(ov_genomic)
  saveViz(upset_plot, output_dir = tempdir(), output_file = "upset_genomic")

  # Save as PNG instead of PDF
 saveViz(upset_plot, format = "png", output_dir = tempdir(), output_file = "upset_example")

  # Save as SVG
 saveViz(venn_plot, format = "svg", output_dir = tempdir(), output_file = "venn_example")
```

today *Today's Date at Package Load Time*

Description

This variable stores the current date (in "yyyymmdd" format) at the time the package is loaded. It is useful for reproducible filenames (e.g., in `saveViz()`), and is automatically set when the package is attached.

Usage

```
today
```

Format

A character string (e.g., "20250624").

Examples

```
# Print the date stored at package load
library(gVenn)
today

# Use it in a filename
paste0("venn_plot_", today, ".pdf")
```

Index