

Package: vennDiagramLab (via r-universe)

May 18, 2026

Type Package

Title Headless Venn Diagram Analysis and Rendering

Version 2.0.5

Description Headless companion to the 'Venn Diagram Lab' web tool (<<https://www.venndiagramlab.org/>>). Build, render, and statistically analyze Venn / 'UpSet' diagrams from 'CSV' / 'TSV' / 'GMT' / 'GMX' inputs. Provides the same 44 SVG models, intersection / 'Jaccard' / hypergeometric statistics, and PDF report layout as the web tool, with byte-equivalent 'TSV' exports (parity-tested against the published Python package). Integrates with 'ggplot2', 'tidygraph', and 'broom'.

License MIT + file LICENSE

Language en-US

URL <https://zoliqua.github.io/Venn-Diagram-Lab/r/>,
<https://github.com/ZoliQua/Venn-Diagram-Lab>,
<https://www.venndiagramlab.org/>

BugReports <https://github.com/ZoliQua/Venn-Diagram-Lab/issues>

Depends R (>= 4.2)

Imports methods, stats, utils, jsonlite, xml2, ggplot2, ComplexUpset, ggraph, tidygraph, grDevices, rsvg, patchwork, gridExtra, BiocGenerics

Suggests testthat (>= 3.0.0), knitr, rmarkdown, rprojroot, pdftools, broom, tibble, dplyr, covr, BiocCheck, lintr, pkgdown

VignetteBuilder knitr

biocViews Visualization, GeneSetEnrichment, Software

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Zoltán Dul [aut, cre] (ORCID: <https://orcid.org/0000-0002-9523-3450>), Márton Ölbei [aut] (ORCID: <https://orcid.org/0000-0002-4903-6237>), N. Shaun B. Thomas [aut], Azeddine Si Ammour [aut] (ORCID: <https://orcid.org/0000-0002-5504-4444>), Attila Csikász-Nagy [aut] (ORCID: <https://orcid.org/0000-0002-2919-5601>)

Maintainer Zoltán Dul <zoltan.dul@gmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-05-18 19:09:41 UTC

RemoteUrl <https://github.com/cran/vennDiagramLab>

RemoteRef HEAD

RemoteSha 9e94544c6bfec9ca8c7b7397b3a37bff3a9a243d

Contents

analyze	3
augment.RegionResult	4
bh_fdr	5
circle_intersection_area	5
compute_pairwise	6
dice	7
effective_universe	7
fold_enrichment	8
generate_proportional_svg	9
geom_venn	10
glance.RegionResult	11
hypergeometric_p_value	12
jaccard	13
list_models	13
list_samples	14
load_csv	14
load_gmt	15
load_gmx	15
load_sample	16
load_tsv	17
overlap_coefficient	17
RegionData-class	18
RegionResult-class	18
render_network	19
render_upset	20
render_venn_svg	21
solve_2set	22
solve_3set	23
statistics	24
StatisticsResult-class	25
tidy.RegionResult	25

to_matrix_tsv	26
to_pdf_report	27
to_region_summary_tsv	28
to_statistics_tsv	29
vdl_version	30
VennDataset-class	30

Index	31
--------------	-----------

analyze	<i>Analyze a Venn diagram dataset</i>
---------	---------------------------------------

Description

Compute the Venn region map for a [`VennDataset-class`] and bind it to a model.

Usage

```
analyze(dataset, model = "auto")
```

Arguments

dataset	A [<code>VennDataset-class</code>] (from one of the <code>load_*</code> functions).
model	Model identifier. <code>"auto"</code> picks the canonical model for the dataset's set count (alphabetical first match), e.g. 4 sets -> <code>'venn-4-set'</code> . <code>"proportional"</code> requests an area-proportional layout (only supports 2-3 sets, added in Phase 3). Otherwise pass an explicit name from <code>[list_models()]</code> .

Value

A [`RegionResult-class`] with the per-region item membership, set sizes, and (lazily) `'statistics(result)'`.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
result@model

ds <- load_sample("dataset_real_cancer_drivers_4")
result <- analyze(ds, model = "auto")
result@model
```

augment.RegionResult *Augment method for RegionResult (broom-compatible)*

Description

Returns one row per item in the dataset's universe, with boolean columns indicating set membership and a 'region_label' column naming the exact region (e.g. "A", "AB", "ABC") the item belongs to. Item ordering follows 'dataset@item_order' (first-seen across all sets, JS Set/Map semantics).

Usage

```
## S3 method for class 'RegionResult'
augment(x, ...)
```

Arguments

x	A ['RegionResult-class'].
...	Unused (broom convention).

Details

Region labels use the package's positional letter convention (A-I), matching the labels in 'RegionResult@regions' and the bundled SVG models, regardless of the dataset's 'set_names'.

Value

A tibble (or data.frame fallback) with 'nrow(out) == length(x@dataset@item_order)' and columns: 'item' (character), one logical column per set (named after the set), 'region_label' (character).

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
if (requireNamespace("broom", quietly = TRUE)) broom::augment(result)

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
broom::augment(result)
```

bh_fdr	<i>Benjamini-Hochberg FDR adjustment</i>
--------	--

Description

Wraps 'stats::p.adjust(p, method = "BH)'. Returns adjusted p-values in the same order as the input. Empty input -> empty output.

Usage

```
bh_fdr(p_values)
```

Arguments

p_values Numeric vector of raw p-values in [0, 1].

Value

Numeric vector of adjusted p-values, same length as input.

Examples

```
bh_fdr(c(0.001, 0.01, 0.05, 0.5))
```

circle_intersection_area	<i>Lens-shaped intersection area of two circles</i>
--------------------------	---

Description

Mirrors 'src/utls/proportionalLayout.ts circleIntersectionArea' (web tool) and Python 'circle_intersection_area' byte-for-byte.

Usage

```
circle_intersection_area(r1, r2, d)
```

Arguments

r1 Radius of circle 1 (positive numeric).
r2 Radius of circle 2 (positive numeric).
d Distance between centers (non-negative numeric).

Value

Numeric: 0 if circles are disjoint, $\pi * \min(r1,r2)^2$ if fully nested, else the lens-shaped intersection area.

Examples

```
circle_intersection_area(1, 1, 1) # ~ 1.228
circle_intersection_area(1, 1, 3) # 0 (disjoint)
```

compute_pairwise	<i>Compute all 5 pairwise statistical tables</i>
------------------	--

Description

Orchestrator that returns a [`StatisticsResult-class`] populated with Jaccard, Dice, Overlap Coefficient, Fold Enrichment (square NxN matrices) plus a long-form hypergeometric table with BH-FDR adjustment.

Usage

```
compute_pairwise(
  set_names,
  inclusive_sizes,
  pairwise_intersections,
  universe_size
)
```

Arguments

`set_names` Ordered character vector of set identifiers (e.g. `c("A", "B", "C")`).

`inclusive_sizes` Named integer vector of inclusive set sizes (`names(inclusive_sizes)` matches `set_names`).

`pairwise_intersections` Named list of pair intersection counts. Keys are `"set_a|set_b"` with `set_a` appearing earlier in `set_names` than `set_b`.

`universe_size` Hypergeometric universe N (population size). Integer ≥ 1 .

Value

A [`StatisticsResult-class`] object.

Examples

```
compute_pairwise(
  set_names = c("A", "B"),
  inclusive_sizes = c(A = 10L, B = 8L),
  pairwise_intersections = list("A|B" = 5L),
  universe_size = 100L
)
```

dice *Sorensen-Dice coefficient*

Description

Computes $2 * |A \text{ intersection } B| / (|A| + |B|)$. Returns 0 if both sets are empty (matches web tool convention).

Usage

```
dice(size_a, size_b, intersection)
```

Arguments

size_a Inclusive size of set A (integer >= 0).
size_b Inclusive size of set B (integer >= 0).
intersection Inclusive intersection size |A intersection B|.

Value

Numeric in [0, 1].

Examples

```
dice(10, 10, 5)
```

effective_universe *Effective hypergeometric universe size for a RegionResult*

Description

Returns the universe N consistent with the web tool. Binary CSV/TSV datasets get 'dataset@universe_size' (= csv.rows.length, includes all-zero rows); aggregated/GMT/GMX datasets fall back to 'length(item_order)' (= lunion of items).

Usage

```
effective_universe(result)

## S4 method for signature 'RegionResult'
effective_universe(result)
```

Arguments

result A ['RegionResult-class'].

Value

Integer, the universe size N.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
effective_universe(result)

ds <- load_sample("dataset_real_cancer_drivers_4")
result <- analyze(ds)
effective_universe(result) # 20000 for binary cancer drivers sample
```

fold_enrichment	<i>Fold enrichment (observed / expected ratio)</i>
-----------------	--

Description

Computes $(k * N) / (K * n)$. Returns 0.0 if any denominator is zero (matches web tool convention).

Usage

```
fold_enrichment(N, K, n, k)
```

Arguments

N	Population size (total items in the universe). Integer ≥ 1 .
K	Number of success states in the population (e.g. inclusive A). Integer ≥ 0 .
n	Number of draws (e.g. inclusive B). Integer ≥ 0 .
k	Observed successes (e.g. A intersection B). Integer ≥ 0 .

Value

Numeric (≥ 0 ; can exceed 1 for over-representation).

Examples

```
fold_enrichment(20000, 138, 581, 126)
```

`generate_proportional_svg`*Generate an area-proportional SVG for a 2- or 3-set RegionResult*

Description

Circle sizes and inter-circle distances are solved analytically (2-set, exact) or by triangulation (3-set, approximate) so that overlap areas match the requested intersection counts. The returned SVG matches the 44-model schema: ShapeA-I, NameA-I, Count_*, CountSUM_*, Bullet* elements are all present and addressable via xml2.

Usage

```
generate_proportional_svg(  
  result,  
  width = .PROP_DEFAULT_WIDTH,  
  height = .PROP_DEFAULT_HEIGHT  
)
```

Arguments

<code>result</code>	A [<code>'RegionResult-class'</code>].
<code>width</code>	Canvas width in pixels (default 600).
<code>height</code>	Canvas height in pixels (default 600).

Value

A `'character'` (length 1) with the raw SVG.

Examples

```
tmp <- tempfile(fileext = ".tsv")  
writeLines(c("Gene\tSetA\tSetB",  
            "GENE1\t1\t0",  
            "GENE2\t1\t1",  
            "GENE3\t0\t1"), tmp)  
ds <- load_tsv(tmp, binary = TRUE)  
res <- analyze(ds, model = "proportional")  
svg <- generate_proportional_svg(res)  
nchar(svg) > 0
```

geom_venn

*Embed a rendered Venn diagram as a ggplot2 layer***Description**

Returns a list of ggplot2 layers that draw ‘data’ (a [‘RegionResult-class’]) as a rasterized Venn diagram on a unit-square coordinate system, ready to compose with other ggplot2 elements (titles, themes, additional annotations).

Usage

```
geom_venn(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  width_px = .GEOM_VENN_DEFAULT_WIDTH
)
```

Arguments

mapping	Accepted for ggplot2 layer-signature consistency. Currently ignored (the Venn diagram is rendered from ‘data’, not from aesthetic mappings). Reserved for a future Stat-based extension.
data	A [‘RegionResult-class’] (required). The Venn diagram to embed.
stat	Accepted for signature consistency; currently ignored.
position	Accepted for signature consistency; currently ignored.
...	Forwarded to ‘ggplot2::annotation_custom()’ (e.g. ‘xmin’, ‘xmax’, ‘ymin’, ‘ymax’ to position the venn on a non-unit coordinate system).
width_px	Raster width in pixels (default 800). Larger values give sharper output at the cost of memory.

Details

This is a NEW capability – the Python package has no equivalent. It uses the same rasterization pipeline as [to_pdf_report()]: render the SVG via [render_venn_svg()], rasterize via ‘rsvg::rsvg_nativeraster()’, and wrap as a ‘grid::rasterGrob()’ inside ‘ggplot2::annotation_custom()’.

Value

A list of ggplot2 layers: an ‘annotation_custom’ carrying the rasterized Venn, a ‘geom_blank’ establishing ‘[0, 1] x [0, 1]’ limits, and a ‘coord_fixed(ratio = 1)’ so the diagram remains square. Note that ‘coord_fixed’ will override any coordinate system the user has already added; add ‘geom_venn()’ before other coord layers to avoid a warning.

Examples

```

ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
if (getRversion() >= "4.6") {
  p <- ggplot2::ggplot() + geom_venn(data = result) + ggplot2::theme_void()
  inherits(p, "ggplot")
}

if (getRversion() >= "4.6") {
  result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
  library(ggplot2)
  ggplot() +
    geom_venn(data = result) +
    theme_void() +
    ggtitle("Cancer driver overlap (4 sources)")
}

```

glance.RegionResult *Glance method for RegionResult (broom-compatible)*

Description

Returns a 1-row tibble summarizing the analysis: number of sets, number of non-empty regions, total unique items, hypergeometric universe size, resolved model name, whether the layout is approximate (proportional 3-set), and the count of statistically significant / highly significant pairs (FDR-adjusted $q < 0.05$ / < 0.001).

Usage

```

## S3 method for class 'RegionResult'
glance(x, ...)

```

Arguments

```

x                    A ['RegionResult-class'].
...                  Unused (broom convention).

```

Value

A 1-row tibble (or data.frame fallback) with columns: `'n_sets'`, `'n_regions'`, `'n_items'`, `'universe_size'`, `'model'`, `'is_approximate'`, `'n_significant'`, `'n_highly_significant'`.

Examples

```

ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
if (requireNamespace("broom", quietly = TRUE)) broom::glance(result)

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
broom::glance(result)

```

hypergeometric_p_value

One-sided hypergeometric p-value (over-representation)

Description

Computes $P(X \geq k)$ where $X \sim \text{Hypergeometric}(N, K, n)$. Returns 1.0 for invalid inputs so the metric is safe to feed into BH-FDR without filtering.

Usage

```
hypergeometric_p_value(N, K, n, k)
```

Arguments

N	Population size (total items in the universe). Integer ≥ 1 .
K	Number of success states in the population (e.g. inclusive A). Integer ≥ 0 .
n	Number of draws (e.g. inclusive B). Integer ≥ 0 .
k	Observed successes (e.g. A intersection B). Integer ≥ 0 .

Details

Maps to R's 'phyper(k - 1, K, N - K, n, lower.tail = FALSE)'. Note that R's phyper parameter convention differs from Python's scipy: R uses 'm' for success-in-population and 'n' for failure-in-population ($= N - K$), where Python uses 'N' for total population.

Value

Numeric in [0, 1].

Examples

```
hypergeometric_p_value(20000, 138, 581, 126)
```

jaccard	<i>Jaccard similarity index</i>
---------	---------------------------------

Description

Computes $|A \cap B| / |A \cup B|$. Matches the web tool's convention of returning 0 when both sets are empty (NaN-safe).

Usage

```
jaccard(size_a, size_b, intersection)
```

Arguments

size_a	Inclusive size of set A (integer ≥ 0).
size_b	Inclusive size of set B (integer ≥ 0).
intersection	Inclusive intersection size $ A \cap B $.

Value

Numeric in $[0, 1]$.

Examples

```
jaccard(10, 10, 5)
jaccard(0, 0, 0)
```

list_models	<i>List all bundled Venn diagram models</i>
-------------	---

Description

Returns metadata for the 44 bundled SVG model templates plus the 'proportional' synthetic generator (added in Phase 3). Read from JSON region files in 'inst/extdata/models/json/'.

Usage

```
list_models()
```

Value

A 'data.frame' with columns 'name' (filename stem), 'set_count' (2-9), and 'display_name' (from the JSON 'name' field). Sorted by '(set_count, name)'.

Examples

```
head(list_models())
```

list_samples	<i>List bundled sample dataset names</i>
--------------	--

Description

Returns the names of the 5 bundled sample datasets, sorted alphabetically. Use [load_sample()] to load one.

Usage

```
list_samples()
```

Value

Character vector of 5 sample identifiers.

Examples

```
list_samples()
```

load_csv	<i>Load a delimited file (CSV/TSV) into a ['VennDataset-class']</i>
----------	---

Description

Supports two layouts: * Binary mode (default): one row per item, with 0/1 columns marking membership in each set. The first 'prefix_cols' columns are item metadata; remaining columns are sets. * Aggregated mode ('binary = FALSE'): each column is a set, and cells contain item identifiers. Empty cells are ignored.

Usage

```
load_csv(path, binary = TRUE, delimiter = NULL, prefix_cols = 1L)
```

Arguments

path	Path to the file.
binary	'TRUE' for binary 0/1 mode (default), 'FALSE' for aggregated.
delimiter	Explicit delimiter override. 'NULL' auto-detects from ',', ';', tab, and space.
prefix_cols	Number of leading metadata columns in binary mode (default 1). Ignored when 'binary = FALSE'.

Value

A ['VennDataset-class'].

Examples

```
tmp <- tempfile(fileext = ".csv")
writeLines(c("Gene,SetA,SetB", "G1,1,0", "G2,1,1", "G3,0,1"), tmp)
ds <- load_csv(tmp, binary = TRUE)
ds@set_names
```

load_gmt	<i>Load a GMT (Gene Matrix Transposed) file into a ['VennDataset-class']</i>
----------	--

Description

Each line is one set: 'set_name<TAB>description<TAB>item1<TAB>item2<TAB>...'. Lines with fewer than 3 tab-separated columns or empty set names are skipped.

Usage

```
load_gmt(path)
```

Arguments

path Path to the .gmt file.

Value

A ['VennDataset-class'].

Examples

```
tmp <- tempfile(fileext = ".gmt")
writeLines(c("SetA\tdesc\tGENE1\tGENE2\tGENE3",
            "SetB\tdesc\tGENE2\tGENE3\tGENE4"), tmp)
ds <- load_gmt(tmp)
ds@set_names
```

load_gmx	<i>Load a GMX file (transposed GMT) into a ['VennDataset-class']</i>
----------	--

Description

Row 0 = set names, row 1 = descriptions, rows 2+ = items column-aligned.

Usage

```
load_gmx(path)
```

Arguments

path Path to the .gmx file.

Value

A [`'VennDataset-class'`].

Examples

```
tmp <- tempfile(fileext = ".gmx")
writeLines(c("SetA\tSetB",
            "desc_A\tdesc_B",
            "GENE1\tGENE2",
            "GENE2\tGENE3"), tmp)
ds <- load_gmx(tmp)
length(ds@items)
```

load_sample	<i>Load a bundled sample dataset by name</i>
-------------	--

Description

Sample datasets ship inside the package under `'inst/extdata/samples/'` and cover biological (cancer drivers, MSigDB pathways) and mock (streaming platforms, gene sets) use cases. Use `[list_samples()]` to enumerate.

Usage

```
load_sample(name)
```

Arguments

name Sample identifier from `[list_samples()]`.

Value

A [`'VennDataset-class'`] with the appropriate format and mode applied.

Examples

```
ds <- load_sample("dataset_mock_gene_sets")
length(ds@set_names)

ds <- load_sample("dataset_real_cancer_drivers_4")
analyze(ds)@model
```

load_tsv	<i>Load a tab-separated file into a ['VennDataset-class']</i>
----------	---

Description

Equivalent to `load_csv(path, binary = binary, delimiter = "\t", prefix_cols = prefix_cols)`.

Usage

```
load_tsv(path, binary = TRUE, prefix_cols = 1L)
```

Arguments

path	Path to the file.
binary	'TRUE' for binary 0/1 mode (default), 'FALSE' for aggregated.
prefix_cols	Number of leading metadata columns in binary mode (default 1). Ignored when 'binary = FALSE'.

Value

A ['VennDataset-class'].

Examples

```
tmp <- tempfile(fileext = ".tsv")
writeLines(c("Gene\\tSetA\\tSetB", "G1\\t1\\t0", "G2\\t1\\t1", "G3\\t0\\t1"), tmp)
ds <- load_tsv(tmp, binary = TRUE)
ds@universe_size
```

overlap_coefficient	<i>Szymkiewicz-Simpson overlap coefficient</i>
---------------------	--

Description

Computes $|A \cap B| / \min(|A|, |B|)$. Useful when one set is much smaller than the other.

Usage

```
overlap_coefficient(size_a, size_b, intersection)
```

Arguments

size_a	Inclusive size of set A (integer ≥ 0).
size_b	Inclusive size of set B (integer ≥ 0).
intersection	Inclusive intersection size $ A \cap B $.

Value

Numeric in [0, 1].

Examples

```
overlap_coefficient(10, 5, 3)
```

RegionData-class *RegionData: one region of a Venn diagram*

Description

Returned as elements of 'RegionResult@regions'. Bitmask convention: bit 'i' set means "in set with index 'i' in 'dataset@set_names'".

Slots

bitmask Region bitmask (1 to $2^n - 1$).

label Human-readable label like "AB" or "ABC".

set_indices Integer vector of 0-based set indices in this region.

set_names Names of the sets in this region.

exclusive_items Items present in exactly these sets.

inclusive_items Items present in at least these sets.

RegionResult-class *RegionResult: result of analyze()*

Description

Bundles the input dataset, chosen model, region map, set sizes, and a lazy ['StatisticsResult-class'] accessible via 'statistics(result)'.

Slots

dataset The input ['VennDataset-class'].

model Resolved model name (e.g. "venn-4-set" or "proportional").

regions Named list keyed by 'as.character(bitmask)', each value a ['RegionData-class']. Only non-empty regions are stored (sparse for high set counts with few overlaps).

set_sizes Named integer vector: set name -> inclusive size.

is_approximate 'TRUE' for the proportional 3-set layout where exact areas can't be achieved with circles.

render_network	<i>Render a force-directed network plot from a RegionResult</i>
----------------	---

Description

Builds a ggraph plot where nodes are sets (sized by inclusive cardinality) and edges are pairwise overlaps (thickness proportional to the chosen metric; blue for FDR-significant edges below ‘significance_threshold’, grey otherwise). Layout uses the deterministic ‘stress’ algorithm from graphlayouts.

Usage

```
render_network(
  result,
  edge_metric = "intersection",
  seed = 42L,
  significance_threshold = 0.05,
  node_color_map = NULL
)
```

Arguments

result	A [<code>'RegionResult-class'</code>].
edge_metric	One of <code>"intersection"</code> , <code>"jaccard"</code> , <code>"fold_enrichment"</code> (capped at 20.0), <code>"overlap_coefficient"</code> .
seed	Retained for API compatibility; currently unused. The ‘stress’ layout algorithm is fully deterministic and does not rely on a random seed.
significance_threshold	FDR p_adjusted threshold below which edges are colored as significant (default 0.05).
node_color_map	Optional named character vector mapping letters (<code>"A"</code> , <code>"B"</code> , ...) to fill hex colors. Unspecified letters default to yellow (<code>"#FFF200"</code>).

Details

Idiomatic R port of Python ‘render_network’ – same parameter contract, but renders via ggraph + tidygraph instead of networkx + matplotlib.

Value

A ‘ggplot’ (ggraph subclass).

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
p <- render_network(result)
inherits(p, "ggplot")

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
p <- render_network(result, edge_metric = "jaccard")
ggplot2::ggsave(tempfile(fileext = ".png"), p, width = 7, height = 7)
```

render_upset

Render an UpSet plot from a RegionResult

Description

Builds a ComplexUpset ggplot showing intersection sizes (top bars), set membership matrix (middle dot grid), and per-set sizes (left bars). Idiomatic R port of Python ‘render_upset’ – same parameter contract, but renders via ComplexUpset (ggplot2) instead of matplotlib (not a 1:1 port).

Usage

```
render_upset(
  result,
  max_columns = 20L,
  sort_by = c("size", "degree"),
  threshold = 0L,
  color_mode = c("depth", "heatmap", "custom"),
  colors = NULL
)
```

Arguments

result	A [<code>‘RegionResult-class’</code>].
max_columns	Maximum number of intersections to display (default 20). Top-N by the active sort.
sort_by	“size” (default – descending) or “degree” (membership count ascending then alphabetical).
threshold	Exclude intersections with size strictly below this value (default 0L = no filter).
color_mode	“depth” (default – viridis on degree), “heatmap” (Reds on size), or “custom” (use the ‘colors’ mapping).
colors	Named character vector mapping intersection LABELS (e.g. “AB”) to fill hex colors when ‘color_mode = “custom”’. Unspecified labels fall back to “#ccc-ccc”.

Value

A ‘ggplot’ object (saveable via ‘ggplot2::ggsave()’).

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
if (getRversion() >= "4.6") {
  p <- render_upset(result)
  inherits(p, "ggplot")
}

if (getRversion() >= "4.6") {
  result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
  p <- render_upset(result, sort_by = "degree", color_mode = "heatmap")
  ggplot2::ggsave(tempfile(fileext = ".png"), p, width = 8, height = 5)
}
```

render_venn_svg

Render a RegionResult onto its model SVG and return the raw SVG string

Description

Loads the bundled SVG template for ‘result@model’ (or the explicit ‘model’ override), walks the DOM via xml2 to overwrite text content (‘Name*’, ‘Count_*’, ‘CountSUM_*’, ‘Title’) and inline ‘fill:’ colors (‘Shape*’, ‘Shape*2’ for Euler extras, ‘Bullet*’), and serializes back to a string.

Usage

```
render_venn_svg(
  result,
  model = NULL,
  set_names = NULL,
  colors = NULL,
  title = NULL,
  show_names = TRUE,
  show_counts = TRUE
)
```

Arguments

result	A [<code>'RegionResult-class'</code>].
model	Optional model id override (filename stem). Default = <code>'result@model'</code> .
set_names	Optional named character vector mapping letters (" <code>A</code> ", " <code>B</code> ", ...) to display names. Unspecified letters fall back to <code>'result@dataset@set_names'</code> .
colors	Optional named character vector mapping letters to fill hex colors. Applies to <code>'BulletX'</code> , <code>'ShapeX'</code> , and <code>'ShapeX2'</code> (Euler extra shapes).
title	Optional title override. If <code>'NULL'</code> , the template's default title text is preserved.
show_names	If <code>'FALSE'</code> , blanks every <code>'NameA-I'</code> element.
show_counts	If <code>'FALSE'</code> , blanks every <code>'Count_*'</code> and <code>'CountSUM_*'</code> element.

Details

For `'model = "proportional"`, delegates to `[generate_proportional_svg()]`.

Mirrors Python `'render_venn_svg'` byte-for-byte except for: (a) the return type is `'character'` instead of an `'SvgImage'` wrapper class; (b) xml2 may emit slightly different whitespace/attribute ordering than lxml. Functional content (text, fill colors, structure) is identical.

Value

A `'character'` (length 1) with the raw SVG.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
svg <- render_venn_svg(result)
nchar(svg) > 0

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
svg <- render_venn_svg(result)
nchar(svg) > 0
```

solve_2set

Area-proportional 2-set circle layout

Description

Solves for two circles whose areas equal `'a_only + ab'` and `'b_only + ab'` and whose intersection area equals `'ab'`, by analytical bisection on the inter-center distance. Always exact (returns `'is_approximate = FALSE'`).

Usage

```
solve_2set(a_only, b_only, ab)
```

Arguments

a_only Items in A only (integer ≥ 0).
b_only Items in B only (integer ≥ 0).
ab Items in A intersection B (integer ≥ 0).

Details

Mirrors Python 'solve_2set' byte-for-byte.

Value

A named list with elements:

circles Length-2 list of 'list(cx, cy, r)' named lists.

error Relative error of the achieved area fit (typically $< 1e-4$).

is_approximate Always 'FALSE' for 2-set.

Examples

```
solve_2set(a_only = 30L, b_only = 30L, ab = 10L)
```

solve_3set	<i>Area-proportional 3-set circle layout (Wilkinson 2012-style triangulation)</i>
------------	---

Description

Computes pairwise inter-center distances via bisection on the lens intersection area, then places circle C via barycentric triangulation against AB. Always sets 'is_approximate = TRUE' because perfect 3-circle area-proportional fits don't always exist mathematically.

Usage

```
solve_3set(regions)
```

Arguments

regions Named list keyed by 'as.character(bitmask)' (1..7) -> exclusive count. Missing keys are treated as 0.

Details

Mirrors Python 'solve_3set' byte-for-byte (including the 'error = NaN' deliberate sentinel - 3-set fit error is not measured in v0.1).

Value

A named list with elements ‘circles’ (length 3), ‘error’ (always NaN in v0.1), ‘is_approximate’ (always TRUE).

Examples

```
solve_3set(list("1" = 100L, "2" = 80L, "3" = 30L,
               "4" = 60L, "5" = 20L, "6" = 15L, "7" = 5L))
```

 statistics

Lazy pairwise statistics for a RegionResult

Description

Computes (and on subsequent calls re-computes) the [`StatisticsResult-class`] for the pairwise metric tables. R has no built-in ‘cached_property’ equivalent for S4 slots, so this is recomputed each call. Cache externally via ‘stats <- statistics(result)’ if you need to access it many times.

Usage

```
statistics(result)

## S4 method for signature 'RegionResult'
statistics(result)
```

Arguments

result A [`RegionResult-class`].

Value

A [`StatisticsResult-class`].

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
stats <- statistics(result)
stats@jaccard["A", "B"]

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
stats <- statistics(result)
stats@jaccard
stats@hypergeometric
```

 StatisticsResult-class

StatisticsResult: container for pairwise statistical metric tables

Description

Returned by [compute_pairwise()] and (lazily) by 'statistics()' on a 'RegionResult'. Holds five tables:

Slots

jaccard NxN named matrix of Jaccard indices.

dice NxN named matrix of Sorensen-Dice coefficients.

overlap_coefficient NxN named matrix of Szymkiewicz-Simpson overlap coefficients.

fold_enrichment NxN named matrix of fold-enrichment values.

hypergeometric Long-form data.frame (one row per set pair) with columns: set_a, set_b, intersection, expected, p_value, p_adjusted, significant, highly_significant.

 tidy.RegionResult

Tidy method for RegionResult (broom-compatible)

Description

Returns a long-form table with one row per ordered set pair, combining the five pairwise statistical metrics (Jaccard, Dice, overlap coefficient, fold enrichment, hypergeometric p-value + BH-FDR-adjusted q-value). Pair ordering is '(set_a, set_b)' with 'set_a' appearing earlier in 'result@dataset@set_names'.

Usage

```
## S3 method for class 'RegionResult'
tidy(x, ...)
```

Arguments

x A ['RegionResult-class'].

... Unused (broom convention).

Value

A tibble (or data.frame if 'tibble' is not installed) with columns 'set_a', 'set_b', 'intersection', 'expected', 'jaccard', 'dice', 'overlap_coefficient', 'fold_enrichment', 'p_value', 'p_adjusted', 'significant', 'highly_significant'. One row per unordered pair, so 'n*(n-1)/2' rows for an 'n'-set dataset.

Examples

```

ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
if (requireNamespace("broom", quietly = TRUE)) broom::tidy(result)

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
broom::tidy(result)

```

to_matrix_tsv

Write the Item Matrix TSV

Description

Mirrors the React webapp's "Export Matrix" button + Python's 'RegionResult.to_matrix_tsv()' byte-for-byte.

Usage

```

to_matrix_tsv(result, path)

## S4 method for signature 'RegionResult'
to_matrix_tsv(result, path)

```

Arguments

result	A ['RegionResult-class'].
path	Destination file path.

Details

Columns: Item, <SetName1>, <SetName2>, ..., Region. Rows: one per item. Iteration order: mask 1..(2^n - 1); within each mask, items in 'dataset@item_order'.

Value

Invisibly returns 'path'.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
to_matrix_tsv(result, tempfile(fileext = ".tsv"))

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
to_matrix_tsv(result, tempfile(fileext = ".tsv"))
```

to_pdf_report

Compose a multi-page PDF report from a RegionResult

Description

Writes a US Letter landscape PDF with overview, venn+upset, statistics tables, and (by default) network and methodology pages. Each page has a footer with package version, generation timestamp, and page number.

Usage

```
to_pdf_report(
  result,
  path,
  title = NULL,
  include_network = TRUE,
  include_about = TRUE
)
```

Arguments

result	A [<code>'RegionResult-class'</code>].
path	Output PDF file path.
title	Optional title override for the overview page.
include_network	If <code>'TRUE'</code> (default), include the network page.
include_about	If <code>'TRUE'</code> (default), include the methodology page.

Value

Invisibly returns `'NULL'`. The PDF is written to `'path'`.

Examples

```
if (getRversion() >= "4.6") {
  result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
  to_pdf_report(result, tempfile(fileext = ".pdf"))
}
```

to_region_summary_tsv *Write the Region Summary TSV*

Description

Mirrors the React webapp's "Export Region Summary" button + Python's 'RegionResult.to_region_summary_tsv()' byte-for-byte.

Usage

```
to_region_summary_tsv(result, path)

## S4 method for signature 'RegionResult'
to_region_summary_tsv(result, path)
```

Arguments

result	A ['RegionResult-class'].
path	Destination file path.

Details

Columns: Region, Sets, Depth, Exclusive_Count, Inclusive_Count, Exclusive_Pct, Items. Rows: every region ($1..2^n - 1$). Sorted by (Depth ASC, Region label ASC). Items: semicolon-joined, ordered by 'dataset@item_order'. Cells starting with '='/'+'/'-'/'@' (after optional leading whitespace) are escape-prefixed with a single quote.

Value

Invisibly returns 'path'.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
to_region_summary_tsv(result, tempfile(fileext = ".tsv"))
```

```
result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
to_region_summary_tsv(result, tempfile(fileext = ".tsv"))
```

to_statistics_tsv	<i>Write the pairwise Statistics TSV</i>
-------------------	--

Description

Mirrors the React webapp's DataSummaryPanel "Export Statistics" button + Python's 'RegionResult.to_statistics_tsv()' byte-for-byte.

Usage

```
to_statistics_tsv(result, path)

## S4 method for signature 'RegionResult'
to_statistics_tsv(result, path)
```

Arguments

result	A ['RegionResult-class'].
path	Destination file path.

Details

Columns: Set_A, Set_B, Name_A, Name_B, Size_A, Size_B, Intersection, Union, Jaccard, Overlap_Coeff, Dice, Expected, Fold_Enrichment, P_value, FDR, Significant. Float formatting: * Jaccard / Overlap_Coeff / Dice: 4 decimals via [.js_to_fixed()] * Expected: 2 decimals * Fold_Enrichment: 3 decimals * P_value / FDR: scientific (JS toExponential(2)) if '< 0.001', else 6 decimals * Significant: one of "****", "***", "**", "ns" keyed off FDR thresholds (0.001, 0.01, 0.05).

Rows are sorted by P_value ascending (matches the underlying StatisticsResult).

Value

Invisibly returns 'path'.

Examples

```
ds <- methods::new("VennDataset",
  set_names = c("A", "B"),
  items = list(A = c("x", "y"), B = c("y", "z")),
  item_order = c("x", "y", "z"),
  universe_size = 10L, source_path = NULL, format = "csv")
result <- analyze(ds)
to_statistics_tsv(result, tempfile(fileext = ".tsv"))

result <- analyze(load_sample("dataset_real_cancer_drivers_4"))
to_statistics_tsv(result, tempfile(fileext = ".tsv"))
```

vdl_version	<i>Get the vennDiagramLab package version</i>
-------------	---

Description

Returns the installed version of vennDiagramLab as a character string. This trivial function exists so the Phase 0 skeleton has one public export; Phase 1 introduces the real analyze() / load_*() API.

Usage

```
vdl_version()
```

Value

Character string, the package version (e.g. "2.0.0").

Examples

```
vdl_version()
```

VennDataset-class	<i>VennDataset: in-memory representation of a Venn-diagram input</i>
-------------------	--

Description

Returned by the 'load_*()' family and consumed by [analyze()]. Holds the deduplicated set members, first-seen item ordering for byte-equivalent TSV output, and source metadata (path, format, optional hypergeometric universe size).

Slots

`set_names` Ordered character vector of set identifiers (length 2-9).

`items` Named list ('names(items) == set_names') of character vectors, each containing the deduplicated members of the corresponding set.

`item_order` First-seen insertion order of all items across all sets, matching JS Set/Map semantics. Used by TSV writers (Phase 2) for byte-equivalent output to the web tool.

`universe_size` Hypergeometric universe N (population size) from the source file, when known. Binary CSV/TSV loaders set this to the row count (matching the web tool's 'csv.rows.length'); other formats leave it 'NULL', signaling "compute as length(item_order)" downstream.

`source_path` Original file path if loaded from disk; 'NULL' for in-memory datasets.

`format` Source format: one of "csv", "tsv", "gmt", "gmx".

Index

analyze, [3](#)
augment.RegionResult, [4](#)
bh_fdr, [5](#)

circle_intersection_area, [5](#)
compute_pairwise, [6](#)

dice, [7](#)

effective_universe, [7](#)
effective_universe,RegionResult-method
(effective_universe), [7](#)

fold_enrichment, [8](#)

generate_proportional_svg, [9](#)
geom_venn, [10](#)
glance.RegionResult, [11](#)

hypergeometric_p_value, [12](#)

jaccard, [13](#)

list_models, [13](#)
list_samples, [14](#)
load_csv, [14](#)
load_gmt, [15](#)
load_gmx, [15](#)
load_sample, [16](#)
load_tsv, [17](#)

overlap_coefficient, [17](#)

RegionData-class, [18](#)
RegionResult-class, [18](#)
render_network, [19](#)
render_upset, [20](#)
render_venn_svg, [21](#)

solve_2set, [22](#)
solve_3set, [23](#)

statistics, [24](#)
statistics,RegionResult-method
(statistics), [24](#)
StatisticsResult-class, [25](#)

tidy.RegionResult, [25](#)
to_matrix_tsv, [26](#)
to_matrix_tsv,RegionResult-method
(to_matrix_tsv), [26](#)
to_pdf_report, [27](#)
to_region_summary_tsv, [28](#)
to_region_summary_tsv,RegionResult-method
(to_region_summary_tsv), [28](#)
to_statistics_tsv, [29](#)
to_statistics_tsv,RegionResult-method
(to_statistics_tsv), [29](#)

vdl_version, [30](#)
VennDataset-class, [30](#)