Package: radiatR (via r-universe)

June 21, 2026

Type Package

Title Analysis and Visualisation of Headings and Trajectories in Circular Space

Version 0.4.0

Description A complete pipeline for analysing directional data -- headings and trajectories in circular space. Supply a table of angles directly, or reconstruct headings from movement trajectories imported from 20+ tracking tools (EthoVision, DeepLabCut, SLEAP, TRex, ANY-maze, TrackMate, idtracker.ai, Ctrax, and others) via a unified dialect-based loader system, including multi-bodypart pose estimation with likelihood-weighted centroid. Derives per-trial heading directions using multiple rules (ring-crossing, distal point, pose body-axis, velocity-based, and more). Computes circular statistics including mean direction, resultant length, von Mises and wrapped Cauchy parametric fits, circular correlation (circular-linear and circular-circular), and multi-sample hypothesis tests (Watson-Williams, Rayleigh, and others) with multiple-comparison correction. Renders publication-quality ggplot2 radial plots with overlaid rose diagrams, parametric density curves, and non-parametric kernel density estimates. Includes a browser-based graphical interface requiring no R coding via launch_app().

URL https://johnkirwan.github.io/radiatR/, https://github.com/JohnKirwan/radiatR

BugReports https://github.com/JohnKirwan/radiatR/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

LazyDataCompression gzip

Depends R (>= 4.1.0)

Imports rlang, ggplot2, tibble, purrr, circular, methods, utils, stats

Suggests bpDir, dplyr (>= 1.0.0), testthat (>= 3.0.0), knitr, rmarkdown, ggrepel, arrow, data.table, jsonlite, readr, readxl, yaml, withr, writexl, R.matlab, xml2, shiny (>= 1.7.0), bslib (>= 0.5.0), pkgload, png, jpeg, svglite, shinytest2, chromote

Config/testthat/edition 3

VignetteBuilder knitr

Collate 'Tracks.R' 'circular_boxplot.R' 'circular_mapping.R' 'circular_regression.R' 'coords.R' 'circular_plotting.R' 'circular_statistics.R' 'circular_trials.R' 'data.R' 'globals.R' 'headings.R' 'headings_frame.R' 'launch_app.R' 'loaders.R' 'model_selection.R' 'profile_plot.R' 'radiatR-package.R' 'simulate_tracks.R' 'spelling.R' 'time.R' 'track_metrics.R' 'transforms.R'

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-21 21:58:48 UTC

RemoteUrl https://github.com/JohnKirwan/radiatR

RemoteRef HEAD

RemoteSha 6b59df87b738d35ac84789f526574a57b2680822

Contents

.heading_registry — *Build a data frame of arrow segments representing mean direction vectors Length equals resultant_R; angle equals mean_dir*

Description

Build a data frame of arrow segments representing mean direction vectors Length equals resultant_R; angle equals mean_dir

Usage

```
.heading_registry
```

add_angle_rose — *Add a rose diagram of heading angles to a radiate plot*

Description

Draws angular frequency as filled wedge polygons in the Cartesian coordinate space used by `radiate`. Each wedge spans one angular bin; its outer radius is proportional to the proportion (or count) of frames in that bin. The layer can be faceted by passing the same column used in the parent `radiate(panel_by = ...)` call.

Usage

```
add_angle_rose(
  hd,
  bins = 12L,
  angle_col = "heading",
  group_col = NULL,
  scale = 0.4,
  inner_r = 0,
  normalize = TRUE,
  fill = "steelblue",
  colour = NA,
  alpha = 0.5,
  arc_pts = 20L,
  display = NULL,
  axial = FALSE,
  color = NULL
)
```

Arguments

`hd`	Data frame of headings, e.g. from `pose_to_headings` or `derive_headings(..., frame_select = "all")`.
`bins`	Integer; number of equal angular sectors. Default `12`.
`angle_col`	Column containing headings in radians. Default `"heading"`.
`group_col`	Column used for faceting; must match the `panel_by` argument of the parent `radiate()` call.
`scale`	Maximum outer radius of the tallest wedge as a fraction of the unit circle radius. Default `0.4`.
`inner_r`	Inner radius of wedges. Default `0`; set `> 0` for a donut style.
`normalize`	`TRUE` (default) scales wedges by proportion; `FALSE` uses raw counts.
`fill`	Wedge fill colour. Default `"steelblue"`.
`colour`, `color`	Wedge border colour. Default `NA` (no border). `color` is the American-spelling alias.
`alpha`	Opacity. Default `0.5`.
`arc_pts`	Points used to approximate each wedge arc. Default `20L`.
`display`	A ‘circ_display()‘ controlling rotation, matching the parent ‘radiate()‘ plot. Default ‘NULL‘ uses the input’s ‘display‘ attribute when present, otherwise the identity display.
`axial`	Logical; when ‘TRUE‘, mirror each observation to ‘angle_col + pi‘ before estimation, yielding a period-pi (bidirectional) result. Default ‘FALSE‘.

Value

A `geom_polygon` layer that can be added to a `radiate()` plot with `+`.

`add_circ` *Draw a circular guide.*

Description

Creates a list of annotation layers that render a circle with the requested radius and appearance. The returned list can be added directly to a ggplot.

Usage

```
add_circ(
  radius = 1,
  circle_colour = "grey60",
  circle_alpha = 1,
  circle_size = 1,
  linetype = "solid",
  colour = NULL,
```

```
  linewidth = NULL,
  circle_color = NULL,
  color = NULL
)
```

Arguments

`radius`	Radius of the circle, expressed in the same units as the plot coordinates.
`circle_colour`, `circle_color`	Line colour for the circle. ‘circle_color‘ is the American-spelling alias.
`circle_alpha`	Alpha transparency for the circle.
`circle_size`	Line width for the circle.
`linetype`	Line type for the circle.
`colour`, `color`	ggplot-style alias for ‘circle_colour‘. If supplied, takes precedence over ‘circle_colour‘. ‘color‘ is the American-spelling alias.
`linewidth`	ggplot-style alias for ‘circle_size‘. If supplied, takes precedence over ‘circle_size‘.

Value

A list containing a single ggplot annotation layer.

Examples

```
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ(radius = 1)
```

`add_circ_interval` *Render a pre-computed circular interval arc on a radial plot*

Description

Takes a data frame produced by [compute_circ_interval()] (or equivalent) and renders it as a ‘geom_path()‘ arc at radius ‘radius‘. Each row produces one arc. Rows where ‘lower‘ or ‘upper‘ is ‘NA‘ are silently skipped.

Usage

```
add_circ_interval(
  interval_df,
  colour_col = NULL,
  radius = 1.05,
  linewidth = 1.5,
  colour = NULL,
  linetype = "solid",
```

```
  n_theta = 500L,
  axial = FALSE,
  color_col = NULL,
  color = NULL
)
```

Arguments

`interval_df`	Data frame with columns ‘mean_dir‘, ‘lower‘, ‘upper‘ (radians, ‘[-pi, pi]‘), and optionally ‘wraps‘ (logical). Typically the output of [compute_circ_interval()].
`colour_col`, `color_col`	Optional column name to map to the colour aesthetic. Ignored when ‘colour‘ is also supplied. ‘color_col‘ is the American-spelling alias.
`radius`	Radial position of the arc. Default ‘1.05‘.
`linewidth`	Line width. Default ‘1.5‘.
`colour`, `color`	Fixed colour. When ‘NULL‘ (default) and ‘colour_col‘ is set, colour is mapped from that column; when ‘NULL‘ and no ‘colour_col‘, draws in ‘”black”‘. Supplying any colour string always overrides ‘colour_col‘. ‘color‘ is the American-spelling alias.
`linetype`	Line type. Default ‘”solid”‘.
`n_theta`	Number of points along the arc. Default ‘500L‘.
`axial`	Logical. Render the overlay for axial (bidirectional, mod-pi) data via the angle-doubling method: the CI is drawn at both poles of the axis. Default ‘FALSE‘.

Details

For the Bayesian extension, replace ‘lower‘ and ‘upper‘ in the output of [compute_circ_interval()] with credible interval bounds from any model before calling this function: “‘r iv <- compute_circ_interval(hd) iv$lower <- posterior_lower iv$upper <- posterior_upper ggplot() + coord_fixed() + add_circ_interval(iv) “‘

Value

A ‘geom_path()‘ layer.

See Also

[compute_circ_interval()], [add_heading_interval()]

`add_circ_mean` *Render pre-computed circular mean arrows on a radial plot*

Description

Takes a data frame produced by [compute_circ_mean()] and renders each row as a ‘geom_segment()‘ arrow. ‘mean_dir‘ must be in unit-circle convention (0 = East, CCW), as returned by [compute_circ_mean()]. Rows where ‘mean_dir‘ or ‘resultant_R‘ is ‘NA‘ are silently skipped.

Usage

```
add_circ_mean(
  summary_df,
  colour_col = NULL,
  linewidth = 1,
  colour = NULL,
  arrow_length_cm = 0.2,
  axial = FALSE,
  ...,
  color_col = NULL,
  color = NULL
)
```

Arguments

`summary_df`	Data frame with columns ‘mean_dir‘ (UC radians, 0 to 2pi) and ‘resultant_R‘ (0–1). Typically the output of [compute_circ_mean()].
`colour_col`, `color_col`	Optional. Name of a column in ‘summary_df‘ to map to the colour aesthetic. Ignored when ‘colour‘ is also supplied. ‘color_col‘ is the American-spelling alias.
`linewidth`	Line width of the arrow segment. Default ‘1‘.
`colour`, `color`	Fixed colour. When ‘NULL‘ (default) and ‘colour_col‘ is set, colour is mapped from that column; when ‘NULL‘ and no ‘colour_col‘, draws in ‘"black"‘. Supplying any colour string always overrides ‘colour_col‘. ‘color‘ is the American-spelling alias.
`arrow_length_cm`	Arrowhead length in cm. Default ‘0.2‘.
`axial`	Logical. Render the overlay for axial (bidirectional, mod-pi) data: the mean is drawn as a double-headed axis through the centre. Default ‘FALSE‘.
`...`	Additional arguments forwarded to ‘geom_segment‘ (e.g. ‘linetype‘, ‘alpha‘, or a custom ‘arrow‘ spec that overrides the default).

Value

A ‘geom_segment()‘ layer.

See Also

[compute_circ_mean()], [add_heading_arrow()]

`add_circular_boxplot` *Add a circular boxplot layer to a radial plot*

Description

Renders the [circ_boxplot_stats()] summary onto a [radiate()] polar plot in the Buttarazzi et al. (2018) style: a filled box band between the hinges, whisker arcs, short crossbars at the median and hinges, far-out points, and an optional median arrow. For ‘axial = TRUE‘ every element is drawn at both poles of the axis (offsets are computed with mod-pi arithmetic so the box is a short band even when the axis sits on the 0/pi seam). Composable with ‘+‘.

Usage

```
add_circular_boxplot(
  hd,
  angle_col = "heading",
  axial = FALSE,
  radius = 1,
  width = 0.1,
  colour = "black",
  box_fill = "grey80",
  farout_shape = 8,
  show_median_arrow = TRUE,
  linewidth = 0.8,
  n_theta = 200L,
  display = NULL,
  color = NULL
)
```

Arguments

`hd`	A data frame with a column of heading angles in radians (unit-circle convention), or a numeric vector of angles.
`angle_col`	Name of the angle column when ‘hd‘ is a data frame. Default ‘"heading"‘.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi): angles are doubled, the boxplot computed, and locations halved back to ‘[0, pi)‘; the fence multiplier is taken on the doubled data. Default ‘FALSE‘.
`radius`	Perimeter radius for the box/whiskers. Default ‘1‘.
`width`	Radial thickness of the box band. Default ‘0.1‘.
`colour`, `color`	Outline colour for box, whiskers, crossbars, far-out. Default ‘"black"‘. ‘color‘ is the American-spelling alias.
`box_fill`	Fill colour of the box band. Default ‘"grey80"‘.

`farout_shape` Point shape for far-out values. Default ‘8‘ (star).

`show_median_arrow`
Draw a radial arrow at the median. Default ‘TRUE‘.

`linewidth` Line width for arcs/crossbars. Default ‘0.8‘.

`n_theta` Points per arc. Default ‘200‘.

`display` A [circ_display()] object; when ‘NULL‘ (default), taken from ‘attr(hd, ”display”)‘, falling back to ‘circ_display()‘.

Value

A list of ggplot2 layers, or ‘NULL‘ when the boxplot is not drawable (a ‘warning()‘ is emitted with the reason).

References

Buttarazzi, D., Pandolfo, G. & Porzio, G. C. (2018). A boxplot for circular data. *Biometrics* 74(4), 1492–1501. doi:10.1111/biom.12889

See Also

[circ_boxplot_stats()], [radiate()]

`add_circular_density` *Wrap a pre-computed circular density around the unit circle*

Description

Takes a data frame of ‘(theta, density)‘ pairs – from any source: MLE, kernel estimation, Bayesian posterior predictive, or hand-crafted – and renders it as a radial path (and optionally a filled polygon) around the unit circle boundary. At each angle theta the plotted radius is ‘1 + scale * density(theta) / max(density)‘.

Usage

```
add_circular_density(
  density_df,
  theta_col = "theta",
  density_col = "density",
  colour_col = NULL,
  scale = 0.4,
  colour = "black",
  fill = NA,
  alpha = 0.2,
  linewidth = 0.8,
  ci_fill = "grey70",
  ci_alpha = 0.3,
  color_col = NULL,
  color = NULL
)
```

Arguments

`density_df`	Data frame with columns named by ‘theta_col‘ and ‘density_col‘ (and, optionally, ‘colour_col‘). Each row represents one evaluated angle.
`theta_col`	Name of the angle column (radians, -pi to pi). Default ‘"theta"‘.
`density_col`	Name of the density/count column. Default ‘"density"‘.
`colour_col`, `color_col`	Optional grouping column. When set, separate paths are drawn per group, enabling ggplot2 faceting. ‘color_col‘ is the American-spelling alias.
`scale`	Maximum radial extension above the unit circle. Default ‘0.4‘ (peak at r = 1.4). Density is normalised within each group before scaling.
`colour`, `color`	Fixed line colour used when ‘colour_col‘ is ‘NULL‘. Default ‘"black"‘. ‘color‘ is the American-spelling alias.
`fill`	Colour for the region between the unit circle and the density curve. ‘NA‘ (default) draws no fill.
`alpha`	Alpha transparency for the filled polygon. Default ‘0.2‘.
`linewidth`	Width of the density path. Default ‘0.8‘.
`ci_fill`	Fill colour for the bootstrap confidence band. Only used when ‘density_df‘ contains ‘density_lower‘ and ‘density_upper‘ columns (produced by [compute_circular_density()] with ‘boot_reps > 0‘, or supplied manually). Default ‘"grey70"‘.
`ci_alpha`	Alpha transparency for the confidence band polygon. Default ‘0.3‘.

Details

Because this function only handles rendering, it is agnostic to how the density was produced. To compute from raw headings use [compute_circular_density()] first, or call the convenience wrapper [add_heading_density()] which combines both steps.

For an **axial** (bidirectional) density, do not mirror this pre-computed frame (that would double-count a full-circle density). Instead estimate it with [compute_circular_density()]‘(..., axial = TRUE)‘, which augments the raw sample before estimation, then pass the result here.

Value

A list of one, two, or three ggplot2 layers: optional CI band polygon, optional fill polygon between density and unit circle, and density path line.

See Also

[compute_circular_density()], [add_heading_density()]

Examples

```
library(ggplot2)
# From compute_circular_density:
hd <- data.frame(heading = c(0.2, 0.3, 0.4, 0.5, -0.1, 0.1, 0.6, 0.2))
dens_df <- compute_circular_density(hd)
ggplot() + coord_fixed() + add_circular_density(dens_df)

# From an external model (e.g. brms posterior predictive):
theta_grid <- seq(-pi, pi, length.out = 200)
external_dens <- data.frame(
  theta   = theta_grid,
  density = exp(-2 * (1 - cos(theta_grid - 0.5)))  # von Mises-like
)
ggplot() + coord_fixed() + add_circular_density(external_dens, fill = "steelblue")
```

---

add_circular_kde *Overlay a non-parametric circular kernel density estimate on a radiate plot*

Description

Estimates the circular density using `density.circular` (no distributional assumptions) and draws it as a closed polygon in the same Cartesian coordinate space used by `radiate` and `add_angle_rose`. Unlike `add_vonmises_density`, this makes no assumption about the shape of the distribution and handles multimodal data naturally.

Usage

```
add_circular_kde(
  hd,
  angle_col = "heading",
  group_col = NULL,
  bw = NULL,
  scale = 0.4,
  inner_r = 0,
  n_pts = 512L,
  kernel = "vonmises",
  colour = "tomato",
  linewidth = 0.8,
  fill = NA,
  alpha = 0.8,
  display = NULL,
  axial = FALSE,
  color = NULL
)
```

Arguments

`hd`	Data frame with a heading column in radians.
`angle_col`	Column name. Default `"heading"`.
`group_col`	Column for faceting; must match `panel_by` in the parent `radiate()` call.
`bw`	Bandwidth (concentration). `NULL` uses `bw.nrd.circular` automatic selection.
`scale`	Maximum outer radius as a fraction of the unit circle. Default `0.4`.
`inner_r`	Inner radius. Default `0`.
`n_pts`	Number of evaluation points. Default `512L`.
`kernel`	Kernel name passed to `density.circular`. Default `"vonmises"` (the kernel shape, not a model assumption).
`colour`, `color`	Outline colour. Default `"tomato"`. `color` is the American-spelling alias.
`linewidth`	Outline width. Default `0.8`.
`fill`	Fill colour. Default `NA` (outline only).
`alpha`	Opacity. Default `0.8`.
`display`	A 'circ_display()' controlling rotation, matching the parent 'radiate()' plot. Default 'NULL' uses the input's 'display' attribute when present, otherwise the identity display.
`axial`	Logical; when 'TRUE', mirror each observation to 'angle_col + pi' before estimation, yielding a period-pi (bidirectional) result. Default 'FALSE'.

Details

The `bw` parameter is a *concentration* parameter (analogous to κ of the von Mises kernel) – larger values produce a sharper, data-following estimate; smaller values over-smooth towards uniform. `NULL` (default) selects bandwidth automatically via `bw.nrd.circular`.

Value

A `geom_polygon` layer, or `NULL` if estimation fails.

`add_critical_r` *Add a critical resultant-length circle to a radiate plot*

Description

Draws an inner reference circle at the *critical mean resultant length* – the smallest $\bar{R}$ at which a circular significance test reaches `alpha` for the given sample size. If a group's mean-direction arrow (`add_heading_arrow`) extends beyond this circle, that group's headings are significantly directed at the `alpha` level. This is the convention used by Oriana and similar circular-statistics software.

Usage

```
add_critical_r(
  hd,
  alpha = 0.05,
  test = c("rayleigh", "vtest"),
  angle_col = "heading",
  group_col = NULL,
  per_group = FALSE,
  colour = "firebrick",
  colour_by_group = TRUE,
  linetype = "dashed",
  linewidth = 0.6,
  n_pts = 200L,
  color = NULL,
  color_by_group = NULL
)
```

Arguments

`hd`	Data frame of headings with a heading column (radians).
`alpha`	Significance level. Default `0.05`.
`test`	`"rayleigh"` (default) or `"vtest"`.
`angle_col`	Heading column name. Default `"heading"`.
`group_col`	Column identifying groups. `NULL` pools all rows.
`per_group`	Logical. When `group_col` is set but the plot is not faceted, draw one circle per group (`TRUE`) or a single conservative circle (`FALSE`, default). Ignored when faceting (always per panel).
`colour`, `color`	Circle colour. Default `"firebrick"`. When `per_group = TRUE` and `colour_by_group = TRUE` this is overridden by the colour scale. `color` is the American-spelling alias.
`colour_by_group`, `color_by_group`	Logical. When `per_group = TRUE`, map each circle's colour to its group (`TRUE`, default) or draw every circle in the fixed `colour` while still attaching the group column so the circles facet (`FALSE`). Use `FALSE` to keep per-panel circles a single colour without injecting the grouping levels into the parent plot's colour scale. Ignored unless `per_group = TRUE`. `color_by_group` is the American-spelling alias.
`linetype`	Line type. Default `"dashed"`.
`linewidth`	Line width. Default `0.6`.
`n_pts`	Points used to approximate each circle. Default `200L`.

Details

Two tests are supported:

`"rayleigh"` **(default)** Tests against uniformity with no hypothesised direction. Critical value $\bar{R}_{crit} = \sqrt{-\log(\alpha)/n}$ (asymptotic; accurate for $n \geq 10$).

`"vtest"` Tests against a specific direction. Critical value $\bar{R}_{crit} = z_\alpha/\sqrt{2n}$ at its most powerful (when the observed mean direction equals the hypothesised μ_0); the effective threshold rises as the observed direction departs from μ_0, so this circle is a lower bound.

Sample size `n` is taken per group from `hd`. When `group_col` matches the parent `radiate(panel_by = ...)` argument, one circle is drawn per panel at the radius appropriate to that panel's `n`. For groups overlaid in a single panel, set `per_group = TRUE` to draw one circle per group (colour-matched), or `per_group = FALSE` (default) to draw a single conservative circle using the smallest `n` (largest critical radius).

Value

A `geom_path` layer, or `NULL` if no group has `n >= 2`.

See Also

`add_heading_arrow`, `add_circ`

`add_critical_v_line` *Add a V-test significance boundary to a radiate plot*

Description

Draws the decision boundary for the Rayleigh V test against a specified direction `mu0`. Unlike the Rayleigh test (a circle, see `add_critical_r`), the V test privileges one direction, so its boundary is a *straight line* perpendicular to `mu0` at distance $c = z_\alpha/\sqrt{2n}$ from the centre. A mean-direction arrow (`add_heading_arrow`) is V-significant if and only if its tip falls on the far side of this line – equivalently, if its projection onto `mu0` exceeds c.

Usage

```
add_critical_v_line(
  hd,
  mu0,
  alpha = 0.05,
  angle_col = "heading",
  group_col = NULL,
  per_group = FALSE,
  show_region = FALSE,
  colour = "firebrick",
  linetype = "dashed",
  linewidth = 0.6,
  region_fill = "firebrick",
  region_alpha = 0.08,
  axial = FALSE,
  n_pts = 100L,
  color = NULL
)
```

Arguments

hd	Data frame of headings with a heading column (radians).
mu0	Hypothesised direction in radians (unit-circle convention).
alpha	Significance level. Default 0.05.
angle_col	Heading column name. Default "heading".
group_col	Column identifying groups. NULL pools all rows.
per_group	Logical. Draw one boundary per group (TRUE) or a single conservative boundary (FALSE, default). Ignored when faceting, where each panel gets its own boundary.
show_region	Logical; shade the rejection segment. Default FALSE.
colour, color	Line colour. Default "firebrick". color is the American-spelling alias.
linetype	Line type. Default "dashed".
linewidth	Line width. Default 0.6.
region_fill	Fill colour for the rejection region. Default "firebrick".
region_alpha	Fill opacity. Default 0.08.
axial	Logical. Render the V-test boundary for axial (bidirectional, mod-pi) data: the decision chord is mirrored to both poles. Default 'FALSE'.
n_pts	Points approximating the rejection arc. Default 100L.

Details

The line is clipped to the unit circle (drawn as a chord). With show_region = TRUE the circular segment beyond the line – the rejection region – is shaded.

Sample size n is taken per group from hd, with the same options as add_critical_r: per-panel when faceting, per-group when per_group = TRUE, or a single conservative boundary (smallest n, largest c) otherwise.

Value

A list of ggplot2 layers, or NULL if no group has a boundary inside the unit circle.

See Also

add_critical_r, add_heading_arrow

`add_heading_arrow` *Compute a circular mean arrow and add it to a radial plot in one step*

Description

Convenience wrapper that calls [compute_circ_mean()] followed by [add_circ_mean()]. Use the two-step form directly when you need to inspect or modify the summary data frame before rendering.

Usage

```
add_heading_arrow(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  display = NULL,
  linewidth = 1,
  colour = NULL,
  arrow_length_cm = 0.2,
  axial = FALSE,
  ...,
  color_col = NULL,
  color = NULL
)
```

Arguments

`headings_df`	Data frame with a column of heading angles in radians. [derive_headings()] sets ‘attr(headings_df, ”angle_convention”)‘ and ‘attr(headings_df, ”coords”)‘ automatically.
`heading_col`	Name of the column containing heading angles. Default ‘”heading”‘.
`colour_col`, `color_col`	Optional. Name of a column to group by. One row is returned per group. The same column maps to colour in [add_circ_mean()]. ‘color_col‘ is the American-spelling alias.
`display`	A [‘circ_display‘] object. When ‘NULL‘ (default), read from ‘attr(headings_df, ”display”)‘, falling back to ‘circ_display()‘.
`linewidth`	Line width of the arrow segment. Default ‘1‘.
`colour`, `color`	Fixed colour. When ‘NULL‘ (default) and ‘colour_col‘ is set, colour is mapped from that column; when ‘NULL‘ and no ‘colour_col‘, draws in ‘”black”‘. Supplying any colour string always overrides ‘colour_col‘. ‘color‘ is the American-spelling alias.
`arrow_length_cm`	Arrowhead length in cm. Default ‘0.2‘.

`axial` Logical. Render the overlay for axial (bidirectional, mod-pi) data: the mean is drawn as a double-headed axis through the centre. Default 'FALSE'.

`...` Additional arguments forwarded to 'geom_segment' (e.g. 'linetype', 'alpha', or a custom 'arrow' spec that overrides the default).

Value

A 'geom_segment()' layer.

See Also

[compute_circ_mean()], [add_circ_mean()]

`add_heading_density` *Compute a circular density and add it to a radial plot in one step*

Description

Convenience wrapper that calls [compute_circular_density()] followed by [add_circular_density()]. Equivalent to: "'r add_circular_density(compute_circular_density(headings_df, heading_col, colour_col, method, ...), colour_col = colour_col, scale = scale, ...) "'

Usage

```
add_heading_density(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  method = c("vonmises", "kernel", "histogram"),
  n_theta = 500L,
  bins = 36L,
  bw = NULL,
  boot_reps = 0L,
  boot_alpha = 0.05,
  scale = 0.4,
  colour = "black",
  fill = NA,
  alpha = 0.2,
  linewidth = 0.8,
  ci_fill = "grey70",
  ci_alpha = 0.3,
  axial = FALSE,
  color_col = NULL,
  color = NULL
)
```

Arguments

`headings_df`	Data frame containing heading angles.
`heading_col`	Name of the heading column (radians). Default '"heading"'.
`colour_col`, `color_col`	Optional grouping column. When set, one density is computed per group and the column is included in the output. 'color_col' is the American-spelling alias.
`method`	Estimation method: '"vonmises"' (default), '"kernel"', or '"histogram"'.
`n_theta`	Number of angular evaluation points for smooth methods. Default '500'.
`bins`	Number of angular bins for the histogram method. Default '36' (10degrees each).
`bw`	Bandwidth passed to [circular::density.circular()]. 'NULL' uses [circular::bw.nrd.circular()].
`boot_reps`	Integer. Number of bootstrap replicates for a '"vonmises"' confidence band. '0' (default) skips the bootstrap. Ignored for '"kernel"' and '"histogram"'.
`boot_alpha`	Significance level for the bootstrap band. Default '0.05' produces a 95% interval.
`scale`	Maximum radial extension above the unit circle. Default '0.4' (peak at r = 1.4). Density is normalised within each group before scaling.
`colour`, `color`	Fixed line colour used when 'colour_col' is 'NULL'. Default '"black"'. 'color' is the American-spelling alias.
`fill`	Colour for the region between the unit circle and the density curve. 'NA' (default) draws no fill.
`alpha`	Alpha transparency for the filled polygon. Default '0.2'.
`linewidth`	Width of the density path. Default '0.8'.
`ci_fill`	Fill colour for the bootstrap confidence band. Only used when 'density_df' contains 'density_lower' and 'density_upper' columns (produced by [compute_circular_density()] with 'boot_reps > 0', or supplied manually). Default '"grey70"'.
`ci_alpha`	Alpha transparency for the confidence band polygon. Default '0.3'.
`axial`	Logical; when 'TRUE', mirror each observation to 'heading_col + pi' before density estimation, producing a period-pi (bidirectional/axial) density. Default 'FALSE'.

Details

Use [compute_circular_density()] + [add_circular_density()] directly when you need to inspect or replace the density values before plotting (e.g. to substitute a Bayesian posterior predictive density from 'brms').

Value

A list of one or two ggplot2 layers.

See Also

[compute_circular_density()], [add_circular_density()]

Examples

```
library(ggplot2)
hd <- data.frame(heading = c(0.2, 0.3, 0.4, 0.5, -0.1, 0.1, 0.6, 0.2))
ggplot() + coord_fixed() + add_heading_density(hd, fill = "steelblue", scale = 0.5)
```

---

add_heading_interval *Compute a circular interval arc and add it to a radial plot in one step*

Description

Convenience wrapper that calls [compute_circ_interval()] followed by [add_circ_interval()]. Use [compute_circ_interval()] + [add_circ_interval()] directly when you need to replace ‘lower‘/‘upper‘ with Bayesian credible interval bounds before rendering.

Usage

```
add_heading_interval(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  display = NULL,
  stat = c("bootstrap_ci", "sd"),
  boot_reps = 1000L,
  boot_alpha = 0.05,
  radius = 1.05,
  linewidth = 1.5,
  colour = NULL,
  linetype = "solid",
  n_theta = 500L,
  axial = FALSE,
  color_col = NULL,
  color = NULL
)
```

Arguments

`headings_df`	Data frame containing heading angles.
`heading_col`	Name of the heading column (radians). Default ‘"heading"‘.
`colour_col`, `color_col`	Optional grouping column. When set, one row is returned per group and the column is preserved in the output. ‘color_col‘ is the American-spelling alias.

`display`	A [‘circ_display‘] object. When ‘NULL‘ (default), read from ‘attr(headings_df, ”display”)‘, falling back to ‘circ_display()‘.
`stat`	Statistic: ‘”bootstrap_ci”‘ (default) or ‘”sd”‘.
`boot_reps`	Integer. Bootstrap replicates for ‘stat = ”bootstrap_ci”‘. Default ‘1000L‘. Ignored when ‘stat = ”sd”‘.
`boot_alpha`	Significance level for the bootstrap CI. Default ‘0.05‘ produces a 95% interval.
`radius`	Radial position of the arc. Default ‘1.05‘.
`linewidth`	Line width. Default ‘1.5‘.
`colour`, `color`	Fixed colour. When ‘NULL‘ (default) and ‘colour_col‘ is set, colour is mapped from that column; when ‘NULL‘ and no ‘colour_col‘, draws in ‘”black”‘. Supplying any colour string always overrides ‘colour_col‘. ‘color‘ is the American-spelling alias.
`linetype`	Line type. Default ‘”solid”‘.
`n_theta`	Number of points along the arc. Default ‘500L‘.
`axial`	Logical. Render the overlay for axial (bidirectional, mod-pi) data via the angle-doubling method: the CI is drawn at both poles of the axis. Default ‘FALSE‘.

Value

A ‘geom_path()‘ layer.

See Also

[compute_circ_interval()], [add_circ_interval()]

`add_heading_points` *Add heading endpoint markers on the unit circle*

Description

Draws a hollow circle at ‘(cos(heading), sin(heading))‘ for each row of a headings data frame, placing one marker per trajectory on the unit-circle boundary at the derived heading direction. The data frame is normally the output of [derive_headings()].

Usage

```
add_heading_points(
  headings_df,
  colour_col = NULL,
  colour = NULL,
  size = 2,
  alpha = 1,
  axial = FALSE,
```

```
  color_col = NULL,
  color = NULL
)
```

Arguments

`headings_df`	Data frame with a ‘heading‘ column (angles in radians).
`colour_col`, `color_col`	Name of a column in ‘headings_df‘ to map to the colour aesthetic. When ‘NULL‘ (default), the value of ‘attr(headings_df, ”colour_col”)‘ is used if set – so heading markers automatically inherit the colour mapping from the associated trajectory plot when that attribute is present. Ignored when ‘colour‘ is supplied. ‘color_col‘ is the American-spelling alias.
`colour`, `color`	Fixed colour string. Overrides ‘colour_col‘ when supplied; when ‘NULL‘ and no ‘colour_col‘ resolves, defaults to ‘”black”‘. ‘color‘ is the American-spelling alias.
`size`	Point size passed to ‘geom_point‘.
`alpha`	Point alpha transparency.
`axial`	Logical; when ‘TRUE‘, draw each observation at both ‘heading‘ and ‘heading + pi‘ (bidirectional/axial display). Default ‘FALSE‘.

Value

A ‘geom_point()‘ layer (shape = 1, hollow circle).

American spellings

Every ‘colour...‘ argument and the ‘assign_colour_*‘ / ‘cycle_colours‘ / ‘hf_colour_col‘ functions accept the American ‘color...‘ spelling as an alias (e.g. ‘color‘, ‘color_col‘, ‘track_color‘). British spelling is canonical; supplying both spellings of a pair is an error.

See Also

[add_heading_vectors()], [derive_headings()]

Examples

```
library(ggplot2)
# headings from a Tracks via derive_headings(ts, rule = "crossing", ...)
hd <- data.frame(id = "A", time = 1, heading = pi / 4)
ggplot() + coord_fixed() + add_heading_points(hd)
ggplot() + coord_fixed() + add_heading_points(hd, colour = "steelblue")
```

`add_heading_vectors` *Add heading vector segments from inner crossing to unit circle*

Description

Draws a segment from the inner-radius crossing position to the heading endpoint on the unit circle for each row of a headings data frame. This visualises the extrapolated vector used to derive the heading and mirrors the dotted-line display in the original millipede tracking workflow.

Usage

```
add_heading_vectors(
  headings_df,
  colour_col = NULL,
  colour = NULL,
  linetype = "dotted",
  axial = FALSE,
  color_col = NULL,
  color = NULL
)
```

Arguments

`headings_df`	Data frame with columns 'heading' (radians), 'x_inner', and 'y_inner'.
`colour_col`, `color_col`	Name of a column in 'headings_df' to map to the colour aesthetic. When 'NULL' (default), the value of 'attr(headings_df, "colour_col")' is used if set – so vectors automatically inherit the colour mapping from the associated trajectory plot when that attribute is present. Ignored when 'colour' is supplied. 'color_col' is the American-spelling alias.
`colour`, `color`	Fixed colour string. Overrides 'colour_col' when supplied; when 'NULL' and no 'colour_col' resolves, defaults to '"black"'. 'color' is the American-spelling alias.
`linetype`	Line type string or integer passed to 'geom_segment'.
`axial`	Logical; when 'TRUE', draw each vector at both 'heading' and 'heading + pi', with the inner start point reflected through the origin. Default 'FALSE'.

Details

Requires columns 'heading', 'x_inner', and 'y_inner', which are present when [derive_headings()] is called with 'rule = "crossing"' and 'return_coords = TRUE'.

Value

A 'geom_segment()' layer.

See Also

[add_heading_points()], [derive_headings()]

Examples

```
library(ggplot2)
hd <- data.frame(id = "A", time = 1, heading = pi / 4,
                 x_inner = 0.15, y_inner = 0.15)
ggplot() + coord_fixed() + add_heading_vectors(hd)
```

---

add_multiple_circles *Add multiple concentric circles to a ggplot object*

Description

This function creates a list of layers for concentric circles with specified radii that can be added to a ggplot object. The function takes optional arguments to customize the appearance of the circles.

Usage

```
add_multiple_circles(
  radii = c(0.25, 0.5, 0.75),
  circle_colour = "grey20",
  circle_alpha = 1,
  circle_size = 0.5,
  circle_color = NULL
)
```

Arguments

`radii`	A vector of radii for the concentric circles (default is c(0.25, 0.5, 0.75))
`circle_colour`, `circle_color`	The colour of the circles (default is "grey20"). 'circle_color' is the American-spelling alias.
`circle_alpha`	The transparency of the circles (default is 1)
`circle_size`	The size of the circle lines (default is 1)

Value

A list of layers for concentric circles

Examples

```
library(ggplot2)
ggplot() + coord_fixed() + add_multiple_circles()
```

add_origin_point *Mark the centre of a radial plot*

Description

Adds a single point at the origin ‘(0, 0)‘ – a centre reference for sparse themes where no crosshairs meet at the middle.

Usage

```
add_origin_point(colour = "grey50", size = 1.5, shape = 16, ..., color = NULL)
```

Arguments

`colour`, `color`	Point colour. Default ‘”grey50”‘. ‘color‘ is the American-spelling alias.
`size`	Point size. Default ‘1.5‘.
`shape`	Point shape. Default ‘16‘ (filled circle).
`...`	Further arguments passed to [ggplot2::geom_point()].

Value

A ‘geom_point()‘ layer.

Examples

```
library(ggplot2)
ggplot() + coord_fixed() + add_circ() + add_origin_point()
```

add_quadrant_lines *Add quadrant lines to a radial plot*

Description

Draws two dashed lines through the centre of the unit circle – one horizontal (0degrees/180degrees) and one vertical (90degrees/270degrees) – dividing the unit circle into four quadrants. The lines extend to the circumference (unit circle).

Usage

```
add_quadrant_lines(
  colour = "grey60",
  linewidth = 0.5,
  linetype = "dashed",
  color = NULL
)
```

Arguments

`colour`, `color`	Line colour. Default ‘"grey60"‘. ‘color‘ is the American-spelling alias.
`linewidth`	Line width. Default ‘0.5‘.
`linetype`	Line type. Default ‘"dashed"‘.

Value

A ‘geom_segment()‘ layer.

Examples

```
library(ggplot2)
ggplot() + coord_fixed() + add_quadrant_lines()
```

`add_radial_grid` *Radial grid layers (the radial analogue of a Cartesian grid)*

Description

Returns a list of layers – an optional filled disc, concentric rings, and radial spokes, in two weights (major/minor) – to compose onto a radial ‘radiate()‘ plot with ‘+‘. The unit boundary (radius 1) is left to the circumference ([add_circ()]); grid rings are interior only.

Usage

```
add_radial_grid(
  rings_major = 0.5,
  rings_minor = c(0.25, 0.75),
  spokes_major = 4L,
  spokes_minor = 4L,
  colour = "grey92",
  colour_minor = NULL,
  linewidth = 0.5,
  linewidth_minor = NULL,
  linetype = "solid",
  disc_fill = NA,
  origin = FALSE,
  origin_colour = "grey50",
  origin_size = 1.5,
  n_pts = 200L,
  color = NULL,
  color_minor = NULL,
  origin_color = NULL
)
```

Arguments

`rings_major`, `rings_minor`
: Numeric radii (‘<1‘) of the major and minor rings. Defaults ‘0.5‘ and ‘c(0.25, 0.75)‘.

`spokes_major`, `spokes_minor`
: Number of evenly spaced spokes for each weight. ‘4‘ major gives the quadrant crosshairs; ‘4‘ minor interleaves them as 45-degree diagonals. Minor spokes are offset from major by half the major spacing (‘pi / spokes_major‘), so their placement is defined relative to the major count.

`colour`, `colour_minor`, `color`, `color_minor`
: Spoke/ring colour. ‘colour_minor‘ defaults to ‘colour‘. ‘color‘ and ‘color_minor‘ are the American-spelling aliases.

`linewidth`, `linewidth_minor`
: Line widths. ‘linewidth_minor‘ defaults to ‘0.5 * linewidth‘.

`linetype`
: Line type for the spokes. Default ‘”solid”‘.

`disc_fill`
: Fill colour for the background disc; ‘NA‘ (default) draws no disc.

`origin`
: Logical; add a centre dot ([add_origin_point()]). Default ‘FALSE‘.

`origin_colour`, `origin_size`, `origin_color`
: Centre-dot style. ‘origin_color‘ is the American-spelling alias.

`n_pts`
: Points used to approximate the disc outline. Default ‘200L‘.

Value

A list of ggplot2 layers.

See Also

[add_multiple_circles()], [add_quadrant_lines()], [add_origin_point()]

Examples

```
library(ggplot2)
ggplot() + coord_fixed() +
  add_radial_grid(disc_fill = "grey92", colour = "white") +
  add_circ()
```

`add_stacked_headings` *Add stacked heading dots as a ggplot2 layer*

Description

Places one point per observation at its heading angle, stacking coincident angles radially to avoid overplotting. If `stack_r` is already a column in `data` (from a prior call to `stack_headings`), it is used as-is; otherwise stacking is computed internally.

Usage

```
add_stacked_headings(
  data,
  col = NULL,
  step = 0.025,
  start_sep = 0,
  tol = NULL,
  direction = "inward",
  base_r = 1,
  shade = FALSE,
  shape = FALSE,
  group = NULL,
  colour = "black",
  colour_col = NULL,
  size = 2,
  alpha = 1,
  axial = FALSE,
  ...,
  color = NULL,
  color_col = NULL
)
```

Arguments

`data`	A data frame with an angle column in radians, typically a `headings_frame`.
`col`	Name of the angle column. Defaults to the `heading_col` attribute when `data` is a `headings_frame`.
`step`, `start_sep`, `tol`, `direction`, `base_r`	Passed to `stack_headings` when `stack_r` is absent. See that function for details. `step` sets the gap between dots and `start_sep` offsets the first dot off the reference circle.
`shade`	If `TRUE`, map `stack_n` to the alpha aesthetic (scaled 0.2–1 across the observed range). Overrides the fixed `alpha` argument.
`shape`	Passed to `stack_headings` to request per-observation shape encoding. Shape is also applied when `shape_code` is already a column in `data`. Mapped to ggplot2 shape integers: 1 = hollow, 16 = filled, 21 = filled with ring.
`group`	Optional column name; stack within each group independently (e.g. one stacking per facet). Default `NULL`.
`colour`, `color`	Fixed point colour (ignored when `colour_col` is set). `color` is the American-spelling alias.
`colour_col`, `color_col`	Optional column name to map to the colour aesthetic. `color_col` is the American-spelling alias.
`size`	Point size passed to `geom_point()`.
`alpha`	Fixed alpha. Ignored when `shade = TRUE`.

`axial`	Logical; when 'TRUE', mirror each observation to 'col + pi' before stacking, so the figure reads as bidirectional. Stacking is computed after mirroring, so each antipodal cluster stacks within itself. Default 'FALSE'.
`...`	Additional arguments passed to `ggplot2::geom_point()`.

Value

A `geom_point()` layer.

See Also

`headings_frame`, `stack_headings`, `add_heading_points`

---

add_ticks *Create evenly spaced radial tick marks.*

Description

Generates a 'geom_segment()' layer containing 'n' evenly spaced tick marks around the unit circle, each spanning a radial distance of 'length' straddling radius 1. The layer can be added to any ggplot.

Usage

```
add_ticks(
  colour = "black",
  linewidth = 0.5,
  length = 0.1,
  n = 8L,
  color = NULL
)
```

Arguments

`colour`, `color`	Tick colour. Default '"black"'. 'color' is the American-spelling alias.
`linewidth`	Tick line width. Default '0.5'.
`length`	Radial length of each tick, in data units. Default '0.1'.
`n`	Number of evenly spaced ticks. Default '8L'.

Value

A 'geom_segment()' layer.

Examples

```
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_ticks()
```

`add_vonmises_density` *Overlay a fitted von Mises density curve on a radiate plot*

Description

Evaluates the von Mises probability density on a fine angular grid and draws it as a closed polygon in the same Cartesian coordinate space used by `radiate` and `add_angle_rose`. The curve peaks at `scale` so the two layers align when given matching `scale` values.

Usage

```
add_vonmises_density(
  fit,
  scale = 0.4,
  inner_r = 0,
  group_col = NULL,
  n_pts = 360L,
  colour = "steelblue",
  linewidth = 0.8,
  fill = NA,
  alpha = 0.8,
  display = NULL,
  axial = FALSE,
  color = NULL
)
```

Arguments

`fit`	Data frame from `vonmises_fit`, containing at least `mu` and `kappa` columns.
`scale`	Maximum outer radius as a fraction of the unit circle. Default `0.4` matches `add_angle_rose`.
`inner_r`	Inner radius; default `0`.
`group_col`	Column in `fit` for faceting; must match the `panel_by` argument of the parent `radiate()` call.
`n_pts`	Angular evaluation points. Default `360L`.
`colour`, `color`	Outline colour. Default `"steelblue"`. `color` is the American-spelling alias.
`linewidth`	Outline width. Default `0.8`.
`fill`	Fill colour. Default `NA` (outline only).
`alpha`	Opacity. Default `0.8`.
`display`	A ‘circ_display()‘ controlling rotation, matching the parent ‘radiate()‘ plot. Default ‘NULL‘ uses the input’s ‘display‘ attribute when present, otherwise the identity display.
`axial`	Logical; when ‘TRUE‘, draw the axial (bidirectional) density of an axial fit (from ‘vonmises_fit(axial = TRUE)‘): the curve is evaluated on doubled angles, giving two equal peaks at ‘mu‘ and ‘mu + pi‘. Default ‘FALSE‘.

Value

A `geom_polygon` layer, or `NULL` if `fit` is all `NA`.

`add_wrappedcauchy_density` *Overlay a fitted wrapped Cauchy density curve on a radiate plot*

Description

Evaluates `dwrappedcauchy` on a fine angular grid and draws it as a closed polygon in the same Cartesian space as `radiate`. Intended as a visual companion to `add_vonmises_density`: overlaying both curves shows whether the data favour the lighter-tailed von Mises or the heavier-tailed wrapped Cauchy.

Usage

```
add_wrappedcauchy_density(
  fit,
  scale = 0.4,
  inner_r = 0,
  group_col = NULL,
  n_pts = 360L,
  colour = "darkorange",
  linewidth = 0.8,
  fill = NA,
  alpha = 0.8,
  display = NULL,
  axial = FALSE,
  color = NULL
)
```

Arguments

`fit`	Data frame from `wrappedcauchy_fit`, containing at least `mu` and `rho` columns.
`scale`	Maximum outer radius as a fraction of the unit circle. Default `0.4`.
`inner_r`	Inner radius. Default `0`.
`group_col`	Column for faceting; must match `panel_by` in the parent `radiate()` call.
`n_pts`	Angular evaluation points. Default `360L`.
`colour`, `color`	Outline colour. Default `"darkorange"`. `color` is the American-spelling alias.
`linewidth`	Outline width. Default `0.8`.
`fill`	Fill colour. Default `NA` (outline only).
`alpha`	Opacity. Default `0.8`.

`display`	A ‘circ_display()‘ controlling rotation, matching the parent ‘radiate()‘ plot. Default ‘NULL‘ uses the input’s ‘display‘ attribute when present, otherwise the identity display.
`axial`	Logical; when ‘TRUE‘, draw the axial (bidirectional) density of an axial fit (from ‘wrappedcauchy_fit(axial = TRUE)‘): the curve is evaluated on doubled angles, giving two equal peaks at ‘mu‘ and ‘mu + pi‘. Default ‘FALSE‘.

Details

Default colour is `"darkorange"` to distinguish from `add_vonmises_density` (`"steelblue"`) and `add_circular_kde` (`"tomato"`).

Value

A `geom_polygon` layer, or `NULL` if estimation fails.

See Also

`add_vonmises_density`, `add_circular_kde`

`angular_velocity` *Per-observation angular (turning-rate) velocity for a Tracks*

Description

The signed rate of change of travel direction at each point – how fast, and which way, the trajectory is turning. Positive is counter-clockwise (a left turn). Each interior point’s turn (the angle between its incoming and outgoing step) is divided by the centred time step; the first and last point of every trajectory are ‘NA‘.

Usage

```
angular_velocity(
  ts,
  units = c("radians", "degrees"),
  x_col = ts@cols$x,
  y_col = ts@cols$y
)
```

Arguments

`ts`	A ‘Tracks‘.
`units`	‘"radians"‘ per second (default) or ‘"degrees"‘ per second.
`x_col`, `y_col`	Names of the coordinate columns. Default to the ‘Tracks‘’s recorded x/y columns.

Details

Numeric (frame) time requires a frame rate ([set_frame_rate()]); POSIXct time is used directly. The result is scale-free (an angle), so no distance calibration is applied.

Value

A numeric vector, one value per observation in 'ts@data' order, 'NA' at each trajectory's first and last point.

See Also

[velocity_vector()], [instantaneous_speed()], [frame_rate()]

---

apply_transform *Apply a bespoke transformation to a Tracks*

Description

Applies a user-supplied function to a loaded ['Tracks'] – for example a reference-frame correction or an angular remapping – after loading and before summary or plotting, returning a modified 'Tracks' and recording the step in its ['transform_history']. The function is written against the 'Tracks''s column *roles* ('x@cols'), not hard-coded column names.

Usage

```
apply_transform(x, fn, ..., by = c("trajectory", "all"), step = NULL)
```

Arguments

`x`	A ['Tracks'].
`fn`	A function. With 'by = "trajectory"' (default) it is called once per trajectory as 'fn(df, cols, ...)', where 'df' is that trajectory's rows (all columns, including metadata covariates) and 'cols' is 'x@cols'. With 'by = "all"' it is called once on the whole data frame. It must return a data frame with the same rows (same 'id'/'time'); a transform adjusts values, it does not add or drop trials.
`...`	Further arguments passed to 'fn'.
`by`	Either '"trajectory"' (default; per-trajectory) or '"all"' (whole-frame).
`step`	Label recorded in 'transform_history'. Defaults to the name of 'fn', or '"apply_transform"' for anonymous functions.

Value

A ['Tracks'] with '@data' replaced by the transformed result and one step appended to its 'transform_history'.

See Also

[transform_history()], [log_transform()]

Examples

```
data(cpunctatus)

# Recipe A -- a reference-frame offset (e.g. edge-referenced -> centre-
# referenced stimulus) is a frame change, so use set_reference(), not
# apply_transform(): it re-derives rel_theta/rel_x/rel_y consistently. Because
# rel_theta = abs - reference, offsetting the heading by +half (half the
# stimulus's angular width) is a reference change of (reference - half).
ts     <- set_reference(cpunctatus, 0)        # ensure a reference exists
half   <- (20 / 2) * pi / 180                 # a 20-degree-wide stimulus
newref <- reference(ts)$ref_theta - half      # per-trajectory if widths differ
ts2    <- set_reference(ts, stats::setNames(newref, reference(ts)$id))

# Recipe B -- polarization direction -> axis. Polarized-light e-vectors are
# axial (defined mod pi); doubling folds direction into orientation for axial
# circular statistics. (Doubling remaps the angular space, so rel_x/rel_y are
# not physical positions afterwards -- use it for the angle/stat path.)
direction_to_axis <- function(df, cols) {
  df[[cols$angle]] <- (2 * df[[cols$angle]]) %% (2 * pi)
  df
}
ts_axial <- apply_transform(cpunctatus, direction_to_axis, by = "all",
                            step = "direction_to_axis")
```

`as.data.frame.Tracks` *Coerce a Tracks to a data frame*

Description

Returns the long-form trajectory table held in the Tracks's 'data' slot.

Usage

```
## S3 method for class 'Tracks'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A [Tracks-class] object.
`row.names`, `optional`	Accepted for S3 generic consistency; ignored.
`...`	Ignored.

Details

Registered as an S3 method so that ‘as.data.frame(ts)‘ dispatches correctly from any caller. (A bare S4 method on the base S3 generic is only reached from within the package's own namespace, not from user code using the installed package.)

Value

A data frame: the Tracks's ‘data‘ slot.

`assign_colour_key` *Assign a shared colour-key column to a Tracks or data frame*

Description

Writes a colour-key column (‘into‘, default ‘".colour"‘) keyed on ‘by‘, so the tracks and any overlays drawn on top share one colour scale. Two modes are chosen automatically by cardinality: * **Cycled** – ‘by = "trajectory"‘ (the trajectory id column), or any column with more than ‘n‘ levels: the key is a cycled ‘1:n‘ index ([cycle_colours()]), so a high-cardinality key stays legible. * **Distinct** – a column with ‘n‘ or fewer levels: the key holds the column's raw values (as a factor), so a legend is meaningful.

Usage

```
assign_colour_key(x, by, n = 20, reference = NULL, into = ".colour")

assign_color_key(x, by, n = 20, reference = NULL, into = ".colour")
```

Arguments

`x`	A ‘Tracks‘ or data frame to annotate.
`by`	‘"trajectory"‘ (the trajectory id column) or a grouping column name.
`n`	Colour cap / cycle length (positive integer). Default 20.
`reference`	Optional ‘Tracks‘/frame whose key order to reuse. Default ‘NULL‘ uses ‘x‘ itself.
`into`	Name of the key column to add. Default ‘".colour"‘.

Details

Pass ‘reference‘ (another Tracks or frame sharing the key) to borrow its level order, so a given key value gets the same colour in both – e.g. tracks and their heading markers. If ‘by‘ names a column absent from ‘x‘ but present on ‘reference‘, it is borrowed by matching trajectory id.

Value

‘x‘ with the ‘into‘ column added.

American spellings

Every ‘colour...‘ argument and the ‘assign_colour_*‘ / ‘cycle_colours‘ / ‘hf_colour_col‘ functions accept the American ‘color...‘ spelling as an alias (e.g. ‘color‘, ‘color_col‘, ‘track_color‘). British spelling is canonical; supplying both spellings of a pair is an error.

See Also

[cycle_colours()]

Examples

```
ts <- simulate_tracks(n_points = 10, output = "trajset")
ts <- assign_colour_key(ts, by = "trajectory")
```

`assign_cycle_colours` *Assign cycling colour indices to trajectories*

Description

Creates a factor column that assigns each unique trajectory a colour index in the range ‘1:n‘, cycling back to 1 after every ‘n‘ trajectories. When ‘panel_col‘ is supplied the cycle resets independently within each panel so that trajectory 1 in every panel gets index 1.

Usage

```
assign_cycle_colours(
  data,
  id_col,
  n,
  panel_col = NULL,
  out_col = "cycle_colour"
)

assign_cycle_colors(data, id_col, n, panel_col = NULL, out_col = "cycle_colour")
```

Arguments

`data`	A data frame containing at least ‘id_col‘ and, if supplied, ‘panel_col‘.
`id_col`	Name of the column identifying individual trajectories.
`n`	Number of colours to cycle through (positive integer), or a character vector of colour values whose length determines ‘n‘.
`panel_col`	Optional column name. When set the cycle restarts for each unique value of this column.
`out_col`	Name of the new factor column added to ‘data‘. Default ‘”cycle_colour”‘.

Details

The resulting column can be passed to ‘colour_col‘ in [radiate()], [add_heading_points()], or [add_heading_vectors()] to keep colours consistent across layers.

Value

‘data‘ with an additional factor column named ‘out_col‘ (levels ‘"1"‘ through ‘"n"‘).

Examples

```
df <- data.frame(id = paste0("T", 1:12), panel = rep(c("A","B"), each = 6))
df <- assign_cycle_colours(df, id_col = "id", n = 4, panel_col = "panel")
table(df$panel, df$cycle_colour)
```

---

bin_angles *Snap angles to fixed-width circular bin centres*

Description

Bins angles (in radians) into fixed-width sectors and returns each angle snapped to its bin's centre. Snapping coincident-binned angles to a common value is the standard precursor to a stacked dot plot: feed the result to `stack_headings` (with the default `tol = NULL`) to build clean radial columns.

Usage

```
bin_angles(angles, width, phase = 0)
```

Arguments

`angles`	Numeric vector of angles in radians. `NA` is preserved.
`width`	Bin width in radians; must be a single positive number. For a 5-degree bin use `pi / 36`.
`phase`	Radian location of a bin *centre*. The default `0` places bin centres at `0, width, 2 * width, ...`, so the reference direction sits on a column rather than on a bin boundary. Set `phase = width / 2` to reproduce the edge-aligned bins of `circular::plot.circular` (centres at `width / 2`, `3 * width / 2`, ...). Any phase is allowed – e.g. with `width = pi / 2, phase = pi / 4` the bin boundaries fall on the axes, binning by quadrant.

Value

A numeric vector the same length as `angles`, each value snapped to its bin centre and wrapped to `[0, 2 * pi)`.

See Also

stack_headings, add_stacked_headings

Examples

```
# 5-degree bins centred on the reference direction
bin_angles(c(0.01, 0.10, 0.11), width = pi / 36)
# circular-package style (edge-aligned) bins
bin_angles(c(0.01, 0.10, 0.11), width = pi / 36, phase = pi / 72)
```

---

circ_boxplot_stats *Circular boxplot statistics (Tukey-like, for circular and axial data)*

Description

Computes the five-number summary and fences of a circular boxplot following Buttarazzi, Pandolfo & Porzio (2018): observations are depth-ranked outward from the antimedian, the box spans the central ~50 the median), and the fence multiplier is obtained in closed form from the von Mises quantiles at the sample's estimated concentration (so ~0.7 observations are flagged as far-out under that reference). Pairs with [add_circular_boxplot()] for rendering.

Usage

```
circ_boxplot_stats(hd, angle_col = "heading", axial = FALSE)
```

Arguments

hd	A data frame with a column of heading angles in radians (unit-circle convention), or a numeric vector of angles.
angle_col	Name of the angle column when 'hd' is a data frame. Default '"heading"'.
axial	Logical. Treat the angles as axial (bidirectional, mod-pi): angles are doubled, the boxplot computed, and locations halved back to '[0, pi)'; the fence multiplier is taken on the doubled data. Default 'FALSE'.

Value

A list with 'median', 'antimedian', 'hinges', 'box_arc', 'constant', 'kappa', 'fences', 'whiskers', 'far_out' (all radians, unit-circle convention), 'n', 'axial', 'drawable', and 'reason'. When 'drawable' is 'FALSE' (non-unique median or 'n < 4') the location fields are 'NA'; 'reason' may also carry a non-fatal advisory while 'drawable' remains 'TRUE' (near-uniform data).

Source

Algorithm reimplemented from the **bpDir** package.

References

Buttarazzi, D., Pandolfo, G. & Porzio, G. C. (2018). A boxplot for circular data. *Biometrics* 74(4), 1492–1501. doi:10.1111/biom.12889

See Also

[add_circular_boxplot()], [circ_summarise()], [vonmises_fit()]

`circ_cor` *Circular correlation between headings and a covariate*

Description

Computes the association between a heading (angle) series and either a continuous linear covariate (`x_type = "linear"`, default) or a second set of angles (`x_type = "circular"`).

Usage

```
circ_cor(
  hd,
  x_col,
  angle_col = "heading",
  group_col = NULL,
  x_type = c("linear", "circular"),
  test = TRUE
)
```

Arguments

`hd`	Data frame containing the heading and covariate columns.
`x_col`	Name of the covariate column.
`angle_col`	Heading column in radians. Default `"heading"`.
`group_col`	Column to group by. `NULL` uses all rows.
`x_type`	`"linear"` (default) or `"circular"`.
`test`	Logical; include hypothesis test. Default `TRUE`.

Details

Circular-linear (T-linear association, Mardia and Jupp 2000):

$$r^2 = (r_{cx}^2 + r_{cy}^2 - 2r_{cx}r_{cy}r_{xy})/(1 - r_{xy}^2)$$

where r_{cx}, r_{cy}, and r_{xy} are the Pearson correlations of x with $\cos\theta$ and $\sin\theta$, and of $\cos\theta$ with $\sin\theta$. r lies in $[0, 1]$; the test statistic nr^2 is approximately chi-squared with 2 degrees of freedom under the null. Note: r is unsigned (association strength only, not direction).

Circular-circular (Fisher's ρ, via `cor.circular`): $r \in [-1, 1]$.

Value

Tidy data frame with columns `group_col` (if supplied), `r`, `n`, `type`, and when `test = TRUE` also `statistic`, `df`, `p_value`.

---

circ_dispersion *Per-group circular dispersion statistics for a dense heading series*

Description

Computes circular mean direction, mean resultant length R, circular standard deviation, and sample size for each group. Designed for within-trial summaries from `pose_to_headings` or `derive_headings(..., frame_select = "all")`, but accepts any data frame with an angle column.

Usage

```
circ_dispersion(hd, group_col = NULL, angle_col = "heading", axial = FALSE)
```

Arguments

`hd`	Data frame containing headings in radians.
`group_col`	Column(s) to group by (e.g. `"id"` for per-trial summaries). `NULL` treats the entire data frame as one group.
`angle_col`	Name of the heading column. Default `"heading"`.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: statistics are computed via the angle-doubling method and the mean is reported as an axis in '[0, pi)' radians / '[0, 180)' degrees. Default 'FALSE' (ordinary directional data).

Value

Data frame with columns `group_col` (if supplied), `mean_dir`, `resultant_R`, `circ_sd`, `n`. Circular standard deviation is $\sqrt{-2 \log R}$.

---

circ_display *Circular display convention specification*

Description

Describes how unit-circle radian angles are rendered in plots and tables. Pass to any display function as the 'display' argument.

Usage

```
circ_display(zero = pi/2, clockwise = TRUE, units = c("degrees", "radians"))
```

Arguments

zero	UC angle (radians) that maps to display 0. Default ‘pi/2‘ (geographic North at top — standard compass/clock layout). Use ‘0‘ to put a stimulus that lies at East (positive rel_x) at the top.
clockwise	Logical. ‘TRUE‘ (default) for clockwise-positive angles.
units	‘"degrees"‘ (default) or ‘"radians"‘ for table outputs and degree label annotations.

Value

A ‘circ_display‘ list.

---

circ_model_select *Select among candidate circular models by AICc*

Description

Fits three candidate models to a heading sample and ranks them by the small-sample-corrected Akaike information criterion (AICc): a `uniform` distribution (no preferred direction), a `unimodal` von Mises (one preferred direction), and an `axial` (symmetric bimodal) von Mises (a preferred axis, two equal antipodal modes). Answers whether a sample is best described as uniform, directionally, or axially oriented.

Usage

```
circ_model_select(hd, angle_col = "heading", group_col = NULL)
```

Arguments

hd	Data frame with a heading column in radians.
angle_col	Heading column name. Default `"heading"`.
group_col	Column to group by. `NULL` (default) treats the whole data frame as one sample.

Details

Parameters are estimated with `vonmises_fit` (the axial model via its `axial = TRUE` doubled-angle fit) and likelihoods with the `circular` package densities. The table reports model comparison only; obtain the fitted parameters of a chosen model from `vonmises_fit`.

Value

Tidy data frame, one row per candidate model (per group when `group_col` is supplied), sorted by `AICc` ascending (best first; `NA` last). Columns: `group_col` (if supplied), `model`, `n`, `k` (free parameters), `logLik`, `AIC`, `AICc`, `BIC`, `dAICc` (AICc minus the group minimum), and `weight` (Akaike weight). `AICc`/`weight` are `NA` for a model whose fit failed or when `n - k - 1 <= 0`; remaining weights sum to 1.

References

Burnham, K.P. & Anderson, D.R. (2002). Model Selection and Multimodel Inference, 2nd ed. Springer.

See Also

vonmises_fit, test_uniformity

---

circ_regression *Circular-linear regression of a heading on linear covariates*

Description

Fits the Fisher-Lee circular-linear regression $\theta_i \sim \mathrm{vM}(\mu_0 + 2\arctan(x_i'\beta), \kappa)$ via `lm.circular` (`type = "c-l"`), behind a formula interface. The response is a heading in radians (unit-circle convention); the right-hand side supplies one or more linear covariates (factors and interactions are expanded by `model.matrix`).

Usage

```
circ_regression(data, formula, init = NULL)

## S3 method for class 'circ_regression'
summary(object, conf.level = 0.95, ...)

## S3 method for class 'circ_regression'
predict(object, newdata = NULL, ...)

## S3 method for class 'circ_regression'
fitted(object, ...)

## S3 method for class 'circ_regression'
print(x, ...)
```

Arguments

`data`	A data frame containing the response and predictor columns.
`formula`	A formula ‘heading ~ x1 + x2‘; the LHS is the angle column.
`init`	Optional numeric starting values for the slope coefficients (length = number of predictor columns). Default a vector of zeros.
`object`	A `circ_regression` object.
`conf.level`	Confidence level for the coefficient interval. Default 0.95.
`...`	Unused.
`newdata`	Optional data frame of new covariate values. Default uses the training data.
`x`	A `circ_regression` object.

Value

An S3 object of class `"circ_regression"`. Use `summary()` for a tidy coefficient data frame, `predict()` / `fitted()` for fitted mean angles, and `print()` for a compact report. On non-convergence or too few rows, `converged` is `FALSE` and the coefficients are `NA`.

References

Fisher, N. I. & Lee, A. J. (1992). Regression models for an angular response. *Biometrics* 48, 665-677. Mardia, K. V. & Jupp, P. E. (2000). *Directional Statistics*. Wiley.

See Also

`circ_cor`, `vonmises_fit`, `simulate_tracks`

`circ_summarise` *Tidy circular summary of a grouped data frame*

Description

Computes circular summary statistics from a data frame column of angles. Supports grouped tibbles and an explicit `.by` argument, returning one row per group. Output `mean_dir` is always in radians; `mean_dir_deg` is the same value converted to degrees.

Usage

```
circ_summarise(
  data,
  col,
  units,
  .by = NULL,
  axial = FALSE,
  stats = c("n", "mean_dir", "mean_dir_deg", "resultant_R", "kappa"),
  display = circ_display()
)
```

Arguments

`data`	A data frame or grouped tibble.
`col`	Unquoted or quoted name of the column containing angles.
`units`	Units of the angle column: `"radians"` or `"degrees"`. No default – must be specified explicitly. Values are converted to radians internally before computation; output is always in radians (`mean_dir`) or degrees (`mean_dir_deg`). A warning is issued when the value range appears inconsistent with the declared units (e.g. values > 2pi when `units = "radians"`). Suppress range warnings with `options(radiatR.check_units = FALSE)`.

`.by`	Character vector of grouping column names. Overrides any `group_by()` groups on `data`.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: statistics are computed via the angle-doubling method and the mean is reported as an axis in '[0, pi)' radians / '[0, 180)' degrees. Default 'FALSE' (ordinary directional data).
`stats`	Character vector selecting which statistics to compute. Order determines column order in the output. Valid values: `"n"` (count of valid, non-missing headings), `"n_total"` (group size including missing), `"n_missing"` (excluded, non-finite headings), `"mean_dir"`, `"mean_dir_deg"`, `"resultant_R"`, `"kappa"`. Default: `"n"`, `"mean_dir"`, `"mean_dir_deg"`, `"resultant_R"`, `"kappa"` (`"n_total"`/`"n_missing"` are opt-in).
`display`	A ['circ_display'] object. When supplied, 'mean_dir_deg' is converted using the display convention (clockwise, 'zero' offset). When 'NULL' (default), 'mean_dir_deg' is the raw degree equivalent of the unit-circle radian angle.

Value

An ungrouped `tibble` with group columns first followed by requested stat columns in the order given in `stats`.

Examples

```
hd <- data.frame(heading = c(0, pi/4, pi/2), arc = c("a", "a", "b"))
circ_summarise(hd, heading, units = "radians")
circ_summarise(hd, heading, units = "radians", .by = "arc")
circ_summarise(hd, heading, units = "radians", .by = "arc",
               stats = c("n", "mean_dir"))
```

---

circ_summary *Circular summaries per trajectory*

Description

Computes per-trial or global circular statistics (mean direction, resultant length, concentration) from the step-angle column of a 'Tracks'.

Usage

```
circ_summary(x, w = NULL, by = c("id", "global"), axial = FALSE)

circ_summary(x, w = NULL, by = c("id", "global"), axial = FALSE)

## S4 method for signature 'Tracks'
circ_summary(x, w = NULL, by = c("id", "global"), axial = FALSE)
```

Arguments

x	A [‘Tracks‘] object.
w	Character. Name of a weight column in ‘x@data‘. When ‘NULL‘ (default), all steps are weighted equally.
by	Character. ‘"id"‘ (default) returns one row per trial; ‘"global"‘ pools all observations into a single summary row.
axial	Logical. Treat the angles as axial (bidirectional, mod-pi) data: statistics are computed via the angle-doubling method and the mean is reported as an axis in ‘[0, pi)‘ radians / ‘[0, 180)‘ degrees. Default ‘FALSE‘ (ordinary directional data).

Value

data.frame(id, n, t_start, t_end, mean_dir, resultant_R, kappa)

A ‘data.frame‘ with columns ‘id‘, ‘n‘, ‘t_start‘, ‘t_end‘, ‘mean_dir‘ (radians, unit-circle convention, 0 to 2pi), ‘resultant_R‘ (0–1), and ‘kappa‘ (von Mises concentration; ‘NA‘ when estimation fails).

Examples

```
## Not run:
data(cpunctatus)
circ_summary(cpunctatus, by = "id")
circ_summary(cpunctatus, by = "global")

## End(Not run)
```

---

circ_summary_headings

Circular statistics over derived headings

Description

Derives one heading per trial via ‘derive_headings()‘, then computes circular summary statistics (mean direction, resultant length, concentration) optionally grouped by one or more metadata columns.

Usage

```
circ_summary_headings(
  x,
  rule = c("crossing", "distal", "straight", "origin_mean", "net", "velocity_mean"),
  group_by = "id",
  ...
)
```

Arguments

x	A [‘Tracks‘] object.
rule	Character. Heading derivation rule passed to [derive_headings()]. One of ‘"crossing"‘, ‘"distal"‘, ‘"straight"‘, ‘"origin_mean"‘, ‘"net"‘, or ‘"velocity_mean"‘.
group_by	Character vector of column names used to group headings before summarising. Default ‘"id"‘ returns one row per trial. Use ‘NULL‘ for a single global summary row. Any column carried through by [derive_headings()] (e.g. ‘"arc"‘) is valid.
...	Additional arguments forwarded to [derive_headings()], such as ‘circ0‘, ‘circ1‘, ‘return_coords‘, or ‘coords‘.

Value

A ‘data.frame‘ with grouping columns followed by ‘mean_dir‘ (radians, unit-circle convention, 0 to 2), ‘resultant_R‘ (0–1), ‘kappa‘ (von Mises concentration), and ‘n‘ (number of valid headings in the group).

Examples

```
## Not run:
data(cpunctatus)
# per-trial headings (default)
circ_summary_headings(cpunctatus, rule = "crossing", circ0 = 0.2, circ1 = 0.4)

# per-condition summary (requires an "arc" column carried through)
circ_summary_headings(cpunctatus, rule = "crossing",
                      circ0 = 0.2, circ1 = 0.4,
                      group_by = "arc")

## End(Not run)
```

---

circular_mapping *Circular coordinate utilities*

Description

A collection of helpers for converting between Cartesian coordinates and unit-circle representations, including orientation conversions used throughout the package. Most functions are internal, but the angle wrapping helpers remain exported for backwards compatibility.

`compute_circ_interval` *Compute a circular interval arc from heading angles*

Description

Returns a data frame of arc bounds centred on the circular mean direction. Two built-in statistics are available: a bootstrap confidence interval for the mean direction (‘"bootstrap_ci"‘, via [circular::mle.vonmises.bootstrap.ci()]) and +/-1 circular SD (‘"sd"‘, via [circular::sd.circular()]). The ‘lower‘ and ‘upper‘ columns of the output can be replaced with Bayesian credible interval bounds from any model before passing to [add_circ_interval()].

Usage

```
compute_circ_interval(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  stat = c("bootstrap_ci", "sd"),
  boot_reps = 1000L,
  boot_alpha = 0.05,
  axial = FALSE,
  color_col = NULL
)
```

Arguments

`headings_df`	Data frame containing heading angles.
`heading_col`	Name of the heading column (radians). Default ‘"heading"‘.
`colour_col`, `color_col`	Optional grouping column. When set, one row is returned per group and the column is preserved in the output. ‘color_col‘ is the American-spelling alias.
`stat`	Statistic: ‘"bootstrap_ci"‘ (default) or ‘"sd"‘.
`boot_reps`	Integer. Bootstrap replicates for ‘stat = "bootstrap_ci"‘. Default ‘1000L‘. Ignored when ‘stat = "sd"‘.
`boot_alpha`	Significance level for the bootstrap CI. Default ‘0.05‘ produces a 95% interval.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: the interval is computed via the angle-doubling method and ‘mean_dir‘ is reported as an axis in ‘[0, pi)‘, with the endpoints scaled accordingly. Default ‘FALSE‘ (ordinary directional data).

Value

A data frame with columns ‘mean_dir‘, ‘lower‘, ‘upper‘ (radians, ‘[-pi, pi]‘), and ‘wraps‘ (logical, ‘TRUE‘ when the arc crosses the +/-pi discontinuity). ‘lower‘ and ‘upper‘ are ‘NA‘ when ‘n < 3‘.

See Also

[add_circ_interval()], [add_heading_interval()]

`compute_circ_mean` *Compute circular mean direction and resultant length from a headings data frame*

Description

Computes the circular mean direction and resultant length (R) per group from a headings data frame, typically the output of [derive_headings()]. ‘mean_dir‘ in the returned data frame is **always in unit-circle convention** (0 = East, counterclockwise), regardless of the input convention, making it suitable for direct use in [add_circ_mean()].

Usage

```
compute_circ_mean(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  axial = FALSE,
  color_col = NULL
)
```

Arguments

`headings_df`	Data frame with a column of heading angles in radians. [derive_headings()] sets ‘attr(headings_df, "angle_convention")‘ and ‘attr(headings_df, "coords")‘ automatically.
`heading_col`	Name of the column containing heading angles. Default ‘"heading"‘.
`colour_col`, `color_col`	Optional. Name of a column to group by. One row is returned per group. The same column maps to colour in [add_circ_mean()]. ‘color_col‘ is the American-spelling alias.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: ‘mean_dir‘ is the axis in ‘[0, pi)‘ and ‘resultant_R‘ is the axial resultant length, both via the angle-doubling method. Default ‘FALSE‘.

Value

A data frame with columns ‘mean_dir‘ (unit-circle radians, 0 to 2pi), ‘resultant_R‘ (0–1), and ‘colour_col‘ when supplied. Both are ‘NA‘ when a group contains fewer than 2 finite angles.

See Also

[add_circ_mean()], [add_heading_arrow()]

`compute_circular_density` *Compute a circular density data frame from heading observations*

Description

Evaluates a directional density for a set of heading angles and returns a tidy data frame of ‘(theta, density)‘ pairs. The result can be passed directly to [add_circular_density()] for rendering, or inspected and modified before plotting – for example to replace the ‘density‘ column with a Bayesian posterior predictive density obtained from ‘brms‘ or another modelling package.

Usage

```
compute_circular_density(
  headings_df,
  heading_col = "heading",
  colour_col = NULL,
  method = c("vonmises", "kernel", "histogram"),
  n_theta = 500L,
  bins = 36L,
  bw = NULL,
  boot_reps = 0L,
  boot_alpha = 0.05,
  axial = FALSE,
  color_col = NULL
)
```

Arguments

`headings_df`	Data frame containing heading angles.
`heading_col`	Name of the heading column (radians). Default ‘"heading"‘.
`colour_col`, `color_col`	Optional grouping column. When set, one density is computed per group and the column is included in the output. ‘color_col‘ is the American-spelling alias.
`method`	Estimation method: ‘"vonmises"‘ (default), ‘"kernel"‘, or ‘"histogram"‘.

`n_theta`	Number of angular evaluation points for smooth methods. Default ‘500‘.
`bins`	Number of angular bins for the histogram method. Default ‘36‘ (10degrees each).
`bw`	Bandwidth passed to [circular::density.circular()]. ‘NULL‘ uses [circular::bw.nrd.circular()].
`boot_reps`	Integer. Number of bootstrap replicates for a ‘”vonmises”‘ confidence band. ‘0‘ (default) skips the bootstrap. Ignored for ‘”kernel”‘ and ‘”histogram”‘.
`boot_alpha`	Significance level for the bootstrap band. Default ‘0.05‘ produces a 95% interval.
`axial`	Logical; when ‘TRUE‘, mirror each observation to ‘heading_col + pi‘ before density estimation, producing a period-pi (bidirectional/axial) density. Default ‘FALSE‘.

Details

Three built-in estimation methods are provided:

* ‘”vonmises”‘ – fit a von Mises distribution by MLE ([circular::mle.vonmises()]) and evaluate the fitted density on a regular grid of ‘n_theta‘ angles. Bootstrap confidence bands are available via ‘boot_reps‘. * ‘”kernel”‘ – circular kernel density estimate ([circular::density.circular()]) with bandwidth chosen by [circular::bw.nrd.circular()] unless ‘bw‘ is supplied. * ‘”histogram”‘ – angular bin counts (a circular rose diagram); ‘bins‘ controls the number of bins.

When ‘colour_col‘ is supplied the density is computed independently for each group and the group label is preserved in the output, enabling per-panel use with [radiate()]’s ‘panel_by‘.

When ‘boot_reps > 0‘ and ‘method = ”vonmises”‘, a non-parametric bootstrap is run: ‘boot_reps‘ samples are drawn with replacement, a von Mises MLE is fitted to each, and the density is evaluated on the same grid. The ‘boot_alpha / 2‘ and ‘1 - boot_alpha / 2‘ quantiles across replicates are returned as ‘density_lower‘ and ‘density_upper‘ columns. These can be rendered as a confidence band by [add_circular_density()], or replaced with interval values from a Bayesian model before plotting.

Value

A data frame with columns ‘theta‘ (radians, -pi to pi) and ‘density‘ (non-negative), plus ‘colour_col‘ if supplied. When ‘boot_reps > 0‘ and ‘method = ”vonmises”‘, also includes ‘density_lower‘ and ‘density_upper‘. Suitable for passing to [add_circular_density()].

See Also

[add_circular_density()], [add_heading_density()]

Examples

```
hd <- data.frame(heading = c(0.2, 0.3, 0.4, 0.5, -0.1, 0.1, 0.6, 0.2))
dens_df <- compute_circular_density(hd)
head(dens_df)
```

```
# Bootstrap CI band (vonmises only):
## Not run:
dens_df <- compute_circular_density(hd, boot_reps = 999L)
# density_lower / density_upper can be replaced with Bayesian interval values
# before passing to add_circular_density()

## End(Not run)

# Replace the density column with values from an external model before plotting:
# dens_df$density <- my_bayesian_density(dens_df$theta)
# ggplot() + coord_fixed() + add_circular_density(dens_df)
```

`count_goal_entries` *Count entries into a goal zone for trajectories in a circular field*

Description

For each trial, counts the number of times the trajectory enters a circular zone of radius ‘crossing_radius‘ centred on the goal location. Applicable to any circular-field analysis with a defined goal (e.g. the hidden platform in a water maze, a reward zone in an open-field).

Usage

```
count_goal_entries(
  x,
  target_angle,
  target_radius = 1,
  crossing_radius = 0.15,
  coords = c("absolute", "relative")
)
```

Arguments

`x`	A [‘Tracks‘] object with x/y (or rel_x/rel_y) columns registered.
`target_angle`	Numeric. Radians. Direction of the goal from the origin.
`target_radius`	Numeric. Distance of the goal from the origin. Default ‘1‘ (wall). Together with ‘target_angle‘ gives the goal position: ‘gx = target_radius * cos(target_angle)‘, ‘gy = target_radius * sin(target_angle)‘.
`crossing_radius`	Numeric. Radius of the goal zone in unit-circle coordinates. Default ‘0.15‘ (15% of the radius; roughly a 10 cm platform in a 60 cm pool).
`coords`	Character. ‘”absolute”‘ (default) or ‘”relative”‘. See [zone_dwell()].

Details

An ”entry” is a ‘FALSE -> TRUE‘ transition in the ‘distance < crossing_radius‘ sequence (ordered by time). A trajectory that starts inside the zone on the first frame counts as one entry.

Value

A ‘data.frame‘ with one row per trial: ‘id‘ (character) and ‘n_entries‘ (integer).

See Also

[zone_dwell()]

Examples

```
## Not run:
# Water maze probe trial: former platform at 45 degrees (NE), at wall
entries <- count_goal_entries(ts, target_angle = pi / 4,
                              crossing_radius = 0.15)
# n_entries > 1 indicates memory of the platform location

## End(Not run)
```

---

cpunctatus **Cylindroiulus punctatus* visual orientation trajectory dataset*

Description

Pre-processed ‘Tracks‘ of 235 millipede trajectories from a visual-acuity experiment. *Cylindroiulus punctatus* individuals were placed at the centre of a cylindrical arena under bright, downwelling light, with a dark target of varying angular half-width on the arena wall, and their path tracked to test whether they oriented toward the target (object taxis). Eight stimulus conditions are represented: target half-widths of 5, 10, 15, 20, 30, 40, and 50 degrees, plus a featureless control (‘arc = 0‘, an angular subtense of zero).

Usage

```
cpunctatus
```

Format

A [‘Tracks‘] object (44,331 observations) whose data columns include:

trial_id Character. Unique trial identifier.

frame Integer. Video frame number.

trans_x, trans_y Numeric. Unit-circle Cartesian coordinates.

abs_theta Numeric. Absolute bearing (radians).

rel_theta Numeric. Bearing relative to the target direction.

rel_x, rel_y Numeric. Target-relative unit-circle coordinates.

arc Ordered factor. Target half-width in degrees (‘0‘ < ‘5‘ < ‘10‘ < ‘15‘ < ‘20‘ < ‘30‘ < ‘40‘ < ‘50‘; 0 = control).

type Character. ‘"control"‘ or ‘"stimulus"‘.

individual Character. Animal identifier.

Details

These tracks are a **subset** of the full experiment; see Kirwan & Nilsson (2019) for the complete dataset and trial counts.

Coordinates are normalised to the unit circle (arena radius = 1) and rotated so the target lies in a common reference direction; 'rel_theta', 'rel_x', and 'rel_y' give the target-relative heading and position. The subject identifier is retained in the 'individual' column.

The raw dtrack landmark/track text files for every trial are shipped in 'inst/extdata/tracks/', and the trial manifest in 'inst/extdata/millipede_trials.csv', so the full import pipeline can be reproduced with [get_all_object_pos()]. See 'data-raw/millipede_example.R'.

Source

Behavioural experiment from Kirwan & Nilsson (2019); raw tracks produced with dtrack and normalised with **radiatR**. A subset of the published dataset.

References

Kirwan, J. D., & Nilsson, D.-E. (2019). A millipede compound eye mediating low-resolution vision. *Vision Research*, 165, 36–44. doi:10.1016/j.visres.2019.09.003

`cpunctatus_tracks` *Cylindroiulus punctatus* trajectory tibble

Description

A tidy long-form tibble (44,331 rows) holding the per-frame observations from the same millipede experiment as `cpunctatus`, with the trial condition metadata ('arc', 'type', 'individual') joined onto every frame. Convenient for analyses that prefer a plain data frame to the 'Tracks' container. A subset of the published dataset.

Usage

```
cpunctatus_tracks
```

Format

A tibble with 18 columns including `trial_id`, `frame`, `trans_x`, `trans_y`, `rel_x`, `rel_y`, `abs_theta`, `rel_theta`, `arc`, `type`, and `individual`.

Source

Same experiment as `cpunctatus`.

References

Kirwan, J. D., & Nilsson, D.-E. (2019). A millipede compound eye mediating low-resolution vision. *Vision Research*, 165, 36–44. doi:10.1016/j.visres.2019.09.003

`cycle_colours` *Cycle a bounded set of colour indices over the values of a key*

Description

The order-stable primitive behind [assign_cycle_colours()] (and so [radiate()]'s ‘colour_cycle‘). Maps each value of ‘x‘ to an index in ‘1:n‘, numbering the distinct values by ‘levels‘ and wrapping back to ‘1‘ after every ‘n‘. Passing an explicit ‘levels‘ lets two data frames that share a key (for example tracks and an overlay drawn on top of them) be coloured identically, so a given key value gets the same colour in both.

Usage

```
cycle_colours(x, n, levels = NULL)

cycle_colors(x, n, levels = NULL)
```

Arguments

`x`	A vector of key values (e.g. trajectory ids or a grouping column).
`n`	Number of colours to cycle through (a positive integer).
`levels`	Optional ordering of the distinct key values. Defaults to their order of first appearance in ‘x‘. Supply a shared ordering to colour two frames consistently.

Value

A factor the same length as ‘x‘ with levels ‘"1"‘..‘"n"‘ giving the cycled colour index. ‘NA‘ in ‘x‘ is preserved as ‘NA‘.

See Also

[assign_cycle_colours()], [radiate()]

Examples

```
cycle_colours(c("a", "b", "c", "a"), n = 2)
```

`degree_labs` *Label the four diagonal directions.*

Description

Provides a list of annotation layers that mark 45, 135, 225, and 315 degrees on a unit circle.

Usage

```
degree_labs(
  display = circ_display(),
  colour = "black",
  units = NULL,
  size = 3.88,
  family = "",
  color = NULL
)
```

Arguments

`display`	A [‘circ_display‘] object. Default ‘circ_display()‘. Supplies the label units when ‘units‘ is ‘NULL‘.
`colour`, `color`	Label colour. Default ‘”black”‘. ‘color‘ is the American-spelling alias.
`units`	‘”degrees”‘ (e.g. ‘45°‘) or ‘”radians”‘ (e.g. ‘ /4‘). When ‘NULL‘ (default) the units are taken from ‘display‘.
`size`	Label text size, in mm. Default ‘3.88‘ (ggplot2’s default text size).
`family`	Label font family. Default ‘””‘ (the device default).

Value

A list of ggplot2 annotation layers.

Examples

```
library(ggplot2)
ggplot() +
  coord_fixed() +
  degree_labs()
```

`derive_coords` *Derive polar and reference-relative coordinates from unit-circle position*

Description

Given a unit-circle Cartesian position (‘trans_x‘, ‘trans_y‘) and a reference direction, computes the dependent coordinate columns: the radius, the absolute angle (unit-circle and clock conventions), and the reference-relative angle and Cartesian position. This is the single source of the unit-circle -> polar/relative transformation used across the package.

Usage

```
derive_coords(trans_x, trans_y, reference = 0)
```

Arguments

`trans_x`, `trans_y`	Numeric vectors of unit-circle Cartesian coordinates (same length).
`reference`	Reference direction in unit-circle radians; a scalar applied to all points, or a vector recycled per element. Default ‘0‘ (relative frame equals absolute frame).

Value

A data frame with ‘trans_rho‘, ‘abs_theta_clock‘, ‘abs_theta_unit‘, ‘rel_theta_unit‘, ‘rel_x‘, ‘rel_y‘.

See Also

[set_reference()], [reference()]

Examples

```
derive_coords(c(0.5, -0.3), c(0.2, 0.4), reference = pi / 2)
```

`derive_headings` *Derive heading angle(s) from trajectories using specified rule*

Description

Derive heading angle(s) from trajectories using specified rule

Usage

```
derive_headings(
  x,
  rule = c("crossing", "distal", "straight", "origin_mean", "net", "velocity_mean",
    "velocity_axis", "window_net", "goal_bias", "pca_axis", "ransac_straight",
    "maxspeed_window", "vm_fit", "exit", "entry", "ring_tangent"),
  ...,
  coords = c("absolute", "relative")
)

## S4 method for signature 'Tracks'
derive_headings(
  x,
  rule = c("crossing", "distal", "straight", "origin_mean", "net", "velocity_mean",
    "velocity_axis", "window_net", "goal_bias", "pca_axis", "ransac_straight",
    "maxspeed_window", "vm_fit", "exit", "entry", "ring_tangent"),
  ...,
  first_only = FALSE,
  carry = NULL,
  on_missing = c("warn", "error", "quiet"),
  coords = c("absolute", "relative")
)
```

Arguments

x	Tracks
rule	one of "crossing", "distal", "straight"
...	rule-specific parameters, including 'return_coords' (see below)
coords	Character. Which Cartesian columns to use: '"absolute"' (default, uses 'x'/'y' from 'Tracks@cols') or '"relative"' (uses 'rel_x'/'rel_y'; errors if not registered).
first_only	logical; if TRUE, return only the first matching heading per trajectory
carry	optional character vector of columns from the source data to append via nearest time
on_missing	One of '"warn"' (default), '"error"', or '"quiet"', controlling what happens when a rule produces no heading ('NA') for one or more trials. The 'NA' rows are always retained; the returned object carries 'n_total', 'n_missing', and 'missing_ids' attributes. '"warn"' emits a warning, '"error"' stops, '"quiet"' is silent. Rule-based failures are often non-random (e.g. tracks that never reach the circumference) and can bias circular statistics, so they are surfaced by default.

Details

Passing 'return_coords = TRUE' (via '...', default 'FALSE') attaches the construction coordinates each rule used to derive the heading, in the chosen 'coords' frame: 'crossing' adds 'x_inner'/'y_inner'; 'distal' adds 'x_distal'/'y_distal'; 'net' adds 'x_start'/'y_start'/'x_end'/'y_end';

‘straight‘ adds ‘x_seg0‘/‘y_seg0‘/‘x_seg1‘/‘y_seg1‘ (the run endpoints); ‘pca_axis‘ adds ‘x_centroid‘/‘y_centroid‘/‘axis_x‘/‘axis_y‘ (a unit axis vector). Other rules ignore it.

Value

data.frame with columns id, time (approx), heading (radians, unit-circle convention), plus the rule-specific construction columns above when ‘return_coords = TRUE‘. For some rules there may be multiple headings per id.

`directedness_arrow` *Make mean resultant length arrow*

Description

Computes the circular mean direction and resultant length, returning a ‘geom_segment()‘ layer that can be added to an existing ggplot.

Usage

```
directedness_arrow(
  data,
  angle_col,
  arrow_head_cm = 0.2,
  colour = "gray",
  size = 2,
  color = NULL
)
```

Arguments

`data`	Data frame containing the angle column.
`angle_col`	Column containing angles in radians.
`arrow_head_cm`	Length of the arrowhead in centimetres.
`colour`, `color`	Colour of the arrow. ‘color‘ is the American-spelling alias.
`size`	Width of the arrow segment (applied to the geom’s ‘linewidth‘).

Value

A ‘geom_segment()‘ layer.

`distance_scale` *Distance calibration for a Tracks object*

Description

Attach a physical-distance scale (and optional unit label) to a [Tracks] so path lengths and speeds can be reported in real units. The scale is physical units per unit of the recorded ‘x‘/‘y‘ coordinates; it is applied on demand and the stored coordinates are never altered. radiatR otherwise analyses in normalised (unit-arena) space – this is an optional calibration hook.

Usage

```
distance_scale(x)

## S4 method for signature 'Tracks'
distance_scale(x)

distance_unit(x)

## S4 method for signature 'Tracks'
distance_unit(x)

set_distance_scale(x, scale, unit = NULL)

## S4 method for signature 'Tracks'
set_distance_scale(x, scale, unit = NULL)

calibrate_distance(ts, coord_distance, real_distance, unit = NULL)
```

Arguments

`x`	A [Tracks] object.
`scale`	A single positive number: physical units per coordinate unit.
`unit`	Optional single string, the physical unit label (e.g. ‘"mm"‘).
`ts`	A [Tracks] object (for ‘calibrate_distance()‘).
`coord_distance, real_distance`	For ‘calibrate_distance()‘, a known separation in coordinate units and its real-world length; the scale is ‘real_distance / coord_distance‘.

Value

‘distance_scale()‘/‘distance_unit()‘ the stored values (or ‘NULL‘); ‘set_distance_scale()‘/‘calibrate_distance()‘ the modified ‘Tracks‘.

See Also

[frame_rate()], [track_length()], [track_speed()]

`draw_tracks` *Create geom layers for Cartesian track coordinates*

Description

Create geom layers for Cartesian track coordinates

Usage

```
draw_tracks(data, x_col, y_col, geom = "path", mapping = NULL, ...)
```

Arguments

`data`	Data frame that will be plotted.
`x_col`	Name of the column mapped to the x aesthetic.
`y_col`	Name of the column mapped to the y aesthetic.
`geom`	Either a character vector (‘"path"‘/‘"point"‘) or a ggplot2 geom function (e.g. [ggplot2::geom_path]).
`mapping`	Optional aesthetics created with [ggplot2::aes()].
`...`	Additional arguments passed on to the geom function.

Value

A list containing a single ggplot2 layer.

`dtrack_read` *Read a dtrack trajectory file into a Tracks*

Description

Reads a tab-separated, headerless file produced by dtrack (https://bitbucket.org/jochensmolka/dtrack). The file is expected to have at least three columns: frame number, x coordinate, y coordinate. A fourth confidence/flag column (always `1` in practice) is silently dropped.

Usage

```
dtrack_read(path, normalize_xy = FALSE, ...)
```

Arguments

`path`	Path to a dtrack `_point02.txt` trajectory file.
`normalize_xy`	Logical; passed to [read_tracks()]. Default `FALSE` because dtrack files are in pixel space.
`...`	Additional arguments passed to [read_tracks()].

Value

A `Tracks`.

See Also

[import_tracks()] for discovering dtrack file pairs in a directory.

`elapsed_seconds` *Elapsed time per observation of a Tracks object*

Description

Real elapsed time of each point from its own track's start, computed on demand from the time column and (for frame-indexed time) the [frame_rate()]. POSIXct time is used directly; numeric (frame) time requires a frame rate.

Usage

```
elapsed_seconds(ts, units = c("seconds", "minutes", "milliseconds"))
```

Arguments

`ts`	A [Tracks] object.
`units`	‘"seconds"‘ (default), ‘"minutes"‘, or ‘"milliseconds"‘.

Value

A numeric vector aligned to the object's observations.

See Also

[frame_rate()], [track_duration()]

`fitted_directions` *Fitted mean directions from a circular regression, for plotting*

Description

Shapes the predictions of a [circ_regression()] model into the ‘summary_df‘ that [add_circ_mean()] draws, so a fitted mean-direction (rho) arrow can be drawn for each covariate value and colour-coded by the covariate – showing how the mean heading sweeps with the predictor. The arrow length is the model's implied resultant length `circular::A1(kappa)`, the same for every arrow, so direction and colour carry the signal.

Usage

```
fitted_directions(fit, at = NULL, newdata = NULL, display = NULL)
```

Arguments

`fit`	A ‘circ_regression‘ object.
`at`	Numeric (or factor) values for the model's single right-hand-side variable – a convenience for one-predictor models. Supply exactly one of ‘at‘ or ‘newdata‘.
`newdata`	A data frame of covariate values, one row per arrow (for multi-predictor models, or full control). Supply exactly one of ‘at‘/‘newdata‘.
`display`	A [circ_display()] object stored on the result so the arrows orient with the panel. Default [circ_display()].

Value

A data frame with ‘mean_dir‘ (fitted heading, radians, unit-circle convention), ‘resultant_R‘ (= ‘circular::A1(fit$kappa)‘, constant), and the covariate column(s) from ‘newdata‘, with a ‘display‘ attribute. Pass it to ‘add_circ_mean(colour_col = "<predictor>")‘. A non-converged fit yields ‘NA‘ ‘mean_dir‘ rows, which ‘add_circ_mean()‘ skips.

See Also

[circ_regression()], [add_circ_mean()], [compute_circ_mean()]

`frame_rate` *Frame rate of a Tracks object*

Description

Attach a capture frame rate (frames per second) to a [Tracks] so the time aspect of frame-indexed tracks can be represented in real seconds. Stored in the object's metadata; the time/frame column is not altered.

Usage

```
frame_rate(x)

## S4 method for signature 'Tracks'
frame_rate(x)

set_frame_rate(x, fps)

## S4 method for signature 'Tracks'
set_frame_rate(x, fps)
```

Arguments

`x`	A [Tracks] object.
`fps`	A single positive number, frames per second.

Value

'frame_rate()' the stored fps (or 'NULL'); 'set_frame_rate()' the modified 'Tracks'.

See Also

[elapsed_seconds()], [track_duration()]

`get_all_object_pos` *Aggregate track positions across all videos in a manifest.*

Description

Iterates over the rows of 'file_tbl', reading the paired landmark and track files for each video before computing trial limits and normalised coordinates. The per-trial outputs are combined into a single tibble, and the augmented trial limits are returned alongside it.

Usage

```
get_all_object_pos(landmarks = NULL, track = NULL, file_tbl, track_dir)
```

Arguments

`landmarks`	Optional data frame or ‘Tracks‘ for the first entry. Retained for backwards compatibility; values are overwritten internally.
`track`	Optional data frame or ‘Tracks‘ for the first entry.
`file_tbl`	Tibble produced by [import_tracks()], optionally enriched via [load_tracks()] or [load_tracks2()].
`track_dir`	Directory containing the landmark and track text files.

Value

A ‘Tracks‘ combining the normalised observations for all valid trials in the manifest. The aggregated trial limits are available via ‘meta$trial_limits‘.

`get_tracked_object_pos` *Derive trial-level track positions in polar coordinates.*

Description

Using the trial limits returned by [get_trial_limits()], this helper extracts the corresponding rows from a track data frame or ‘Tracks‘, centres and scales the coordinates, and computes angles in both absolute and reference-relative frames. The function optionally controls how inner/outer radius crossings are selected.

Usage

```
get_tracked_object_pos(
  trial_limits,
  track,
  circ0 = 0.1,
  circ1 = 0.2,
  radius_criterion = c("first_past", "closest")
)
```

Arguments

`trial_limits`	Data frame produced by [get_trial_limits()].
`track`	Data frame or ‘Tracks‘ of Cartesian coordinates for the entire video.
`circ0`	Inner radius threshold (default ‘0.1‘).
`circ1`	Outer radius threshold (default ‘0.2‘).
`radius_criterion`	Strategy for choosing the radius landmarks. ‘”first_past”‘ selects the first point beyond each threshold, while ‘”closest”‘ chooses the closest sample to the specified radius.

Value

A ‘Tracks‘ containing all valid trial observations. The corresponding trial limits (including ‘valid_track‘ flags) are stored in ‘meta$trial_limits‘.

---

get_trial_limits *Summarise per-trial metadata for a single video.*

Description

Uses paired landmark coordinates to determine the temporal bounds of each trial, the origin, and the reference heading. Additional metadata from ‘file_tbl‘ is merged into the result. Accepts landmark and track data either as data frames or as ‘Tracks‘ objects.

Usage

```
get_trial_limits(landmarks, track, file_tbl, vid_num)
```

Arguments

`landmarks`	Data frame or ‘Tracks‘ (two rows per trial) containing frame numbers and landmark coordinates.
`track`	Data frame or ‘Tracks‘ of Cartesian coordinates for all frames in the video.
`file_tbl`	Tibble returned by [import_tracks()] (optionally enriched by [load_tracks()] or [load_tracks2()]).
`vid_num`	Index of the current video within ‘file_tbl‘.

Value

A tibble with one row per trial containing trial limits and reference metadata.

---

gg_traj *Plot trajectories from a Tracks (overlay or faceted)*

Description

Plot trajectories from a Tracks (overlay or faceted)

Usage

```
gg_traj(
  x,
  colour = NULL,
  linetype = NULL,
  alpha = NULL,
  size = 0.6,
  panel_by = NULL,
  coord = c("polar", "cartesian"),
  geom = c("path"),
  thin = 1L,
  ncol = NULL,
  color = NULL
)

## S4 method for signature 'Tracks'
gg_traj(
  x,
  colour = NULL,
  linetype = NULL,
  alpha = NULL,
  size = 0.6,
  panel_by = NULL,
  coord = c("polar", "cartesian"),
  geom = c("path"),
  thin = 1L,
  ncol = NULL,
  color = NULL
)
```

Arguments

`x`	Tracks
`colour`, `color`, `linetype`, `alpha`, `size`	Optional column names (strings) mapped to aesthetics. ‘color‘ is the American-spelling alias for ‘colour‘.
`panel_by`	NULL, a single string, or a character vector of columns to facet by
`coord`	”polar” (unit circle) or ”cartesian”
`geom`	”path” or ”point” (or both, as c(”path”,”point”))
`thin`	Keep every n-th point per id (for very long tables)
`ncol`	Number of facet columns (when faceting)

Value

ggplot object

guess_columns *Guess the role of each column in a track table*

Description

Inspects a data frame's column names and returns the best guess for each 'Tracks' role ('id', 'time', 'x', 'y', 'angle', 'weight'), honouring any explicit 'mapping' overrides. Matching is case-insensitive; 'x'/'y' also match separator-suffixed names such as 'Track1_X'. A role with no match is 'NULL'. This is the same logic [read_tracks()] uses internally; call it to see or pre-fill a column mapping. It does not synthesize missing 'id'/'time' columns (a 'NULL' signals that 'read_tracks()' will apply its single-track / row-order fallback).

Usage

```
guess_columns(data, mapping = list())
```

Arguments

data	A data frame (or anything with 'names()').
mapping	Optional named list of explicit role -> column overrides.

Value

A named list with elements 'id', 'time', 'x', 'y', 'angle', 'weight' (each a column name or 'NULL').

Examples

```
guess_columns(data.frame(Frame = 1:2, Track1_X = 0:1, Track1_Y = 0:1))
```

headings_frame *Construct a headings frame from a data frame of angles*

Description

Validates the angle column, optionally converts degrees to radians, and marks the data frame with class `headings_frame` and attributes that downstream functions (`stack_headings`, `add_stacked_headings`, `radiate`) will use as defaults.

Usage

```
headings_frame(
  data,
  col,
  units,
  angle_convention = "unit_circle",
  coords = "absolute"
)
```

Arguments

`data`	A data frame containing the angle column.
`col`	Unquoted or quoted name of the angle column.
`units`	Units of the angle column: `"radians"` or `"degrees"`. No default — must be specified. Values are converted to radians in place when `"degrees"`.
`angle_convention`	`"unit_circle"` (0 = East, CCW, default) or `"clock"` (0 = North, CW).
`coords`	`"absolute"` (default) or `"relative"`.

Value

A `data.frame` with additional class `"headings_frame"` and attributes `heading_col`, `angle_convention`, `coords`.

See Also

`stack_headings`, `add_stacked_headings`, `radiate`

`hf_accessors` *Read the canonical attributes of a heading frame*

Description

Accessors for the metadata a [headings_frame] carries. They also work on a plain data frame, returning sensible defaults, so any function can read the display convention without assuming the input is classed.

Usage

```
hf_display(x)

hf_heading_col(x)

hf_colour_col(x)

hf_color_col(x)

hf_coords(x)
```

Arguments

`x`	A ‘headings_frame‘ or plain data frame.

Value

‘hf_display()‘ a [circ_display()] object; ‘hf_heading_col()‘ / ‘hf_coords()‘ a string; ‘hf_colour_col()‘ a string or ‘NULL‘.

`import_info` *Import landmark coordinates from text files*

Description

Import landmark coordinates from text files

Usage

```
import_info(filename, cond_cols = NULL, file_tbl = NULL)
```

Arguments

`filename`	Path to the CSV file describing each track/landmark pair.
`cond_cols`	Optional character vector of column names whose values should be concatenated to create a 'cond' column.
`file_tbl`	Optional tibble produced by [import_tracks()]. When supplied, the function verifies that the listed files are present in both sources.

Value

A data frame containing the parsed metadata (and optional 'cond' column).

Examples

```
## Not run:
manifest <- import_info("trials_list.csv", cond_cols = c("type", "arc"))

## End(Not run)
```

`import_tracks` *Discover dtrack (or compatible) landmark/track file pairs in a directory*

Description

Scans `dir` for paired files matching `landmark_suffix` and `track_suffix` and returns a tibble of basenames and paths.

Usage

```
import_tracks(dir, landmark_suffix = NULL, track_suffix = NULL)
```

Arguments

`dir`	Directory to scan. Defaults to the current working directory.
`landmark_suffix`	Suffix identifying landmark files. Default `"_point01.txt"`.
`track_suffix`	Suffix identifying trajectory files. Default `"_point02.txt"`.

Details

The default suffixes match the export naming convention used by dtrack (https://bitbucket.org/jochensmolka/dtrack). In the bundled millipede example data, `_point01` files contain two landmark rows per trial (the origin and stimulus edge on the circumference) and `_point02` files contain the per-frame animal trajectory. This two-file role split is specific to that experiment and is not a general dtrack convention. Use [dtrack_read()] to read an individual trajectory file.

Value

A tibble with columns `basename`, `landmark`, and `track`.

`instantaneous_speed` *Per-observation instantaneous speed for a Tracks*

Description

Instantaneous speed at each point, aligned to the ‘Tracks‘’s rows (the per-observation sibling of [elapsed_seconds()]). Each point carries the speed of the step that ends at it; the first point of every trajectory is ‘NA‘. Speeds come from [step_speed()] using the track’s elapsed time from [elapsed_seconds()].

Usage

```
instantaneous_speed(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

`ts`	A ‘Tracks‘.
`x_col`, `y_col`	Names of the coordinate columns. Default to the ‘Tracks‘’s recorded x/y columns.

Details

Numeric (frame) time requires a frame rate ([set_frame_rate()]); POSIXct time is used directly. With the default coordinate columns the unit is arena-units (radii) per second.

Value

A numeric vector, one value per observation in ‘ts@data‘ order, ‘NA‘ at each trajectory’s first point.

See Also

[step_speed()], [track_speed()], [elapsed_seconds()]

`launch_app` *Launch the radiatR Shiny companion app*

Description

Opens a browser-based graphical interface for uploading tracking data, selecting a heading method, and viewing circular track plots and summary statistics. No R coding is required to use the app.

Usage

```
launch_app(port = NULL, launch_browser = TRUE)
```

Arguments

`port`	Integer; TCP port to listen on. `NULL` (default) lets Shiny pick a free port automatically.
`launch_browser`	Logical; open the system browser automatically (default `TRUE`). Set to `FALSE` when running headless.

Details

The same app directory can be deployed to `https://www.shinyapps.io` or any Shiny-compatible server without modification:

```
rsconnect::deployApp(system.file("app", package = "radiatR"))
```

Value

Called for its side-effect; returns the result of `runApp` invisibly.

`line_circle_intercept`

Find the intercept of a line with the unit circle

Description

Find the intercept of a line with the unit circle

Usage

```
line_circle_intercept(x0, y0, x1, y1)
```

Arguments

`x0`, `y0`	Starting coordinates of the vector.
`x1`, `y1`	Ending coordinates of the vector.

Value

A tibble with the intersection point closest to ‘(x1, y1)‘.

`line_circle_intercept_df`

Intersection helper using track rows

Description

Convenience wrapper that accepts a data frame (or ‘Tracks@data‘) containing coordinates and evaluates the intersection between two rows.

Usage

```
line_circle_intercept_df(df, row_in, row_out)
```

Arguments

`df`	Data frame with ‘x‘ and ‘y‘ columns.
`row_in`, `row_out`	Row indices (or names) identifying the start and end of the vector.

Value

Tibble with ‘x_int‘/‘y_int‘.

line_circle_intercept_traj

Intersection helper for Tracks trajectories

Description

Extracts the first/last rows for a given id/time range and computes the intersection with the unit circle.

Usage

```
line_circle_intercept_traj(traj, id, range)
```

Arguments

`traj`	Tracks
`id`	Identifier of the trajectory
`range`	Numeric index vector (e.g., rows within a trial)

Value

Tibble with ‘x_int‘/‘y_int‘

list_heading_rules

List registered custom heading rules

Description

List registered custom heading rules

Usage

```
list_heading_rules()
```

Value

Sorted character vector of registered rule names.

See Also

`register_heading_rule`

list_loader_dialects *List registered loader dialects*

Description

List registered loader dialects

Usage

```
list_loader_dialects()
```

list_loader_formats *List registered declarative formats*

Description

List registered declarative formats

Usage

```
list_loader_formats()
```

load_manifest *Load trajectories listed in a file table into a Tracks*

Description

This high-level helper combines the file discovery tibble returned by [import_tracks()] with optional metadata from an experiment manifest, reads each track file using [read_tracks()], and merges the results into a single ‘Tracks‘.

Usage

```
load_manifest(
  file_tbl,
  track_dir,
  manifest = NULL,
  manifest_cols = NULL,
  mapping = NULL,
  angle_unit = c("radians", "degrees", "auto"),
  time_type = c("auto", "posix", "seconds", "frames"),
  tz = "UTC",
  fps = NULL,
  normalize_xy = TRUE,
```

```
  dialect = NULL,
  keep = NULL,
  drop = NULL,
  ...
)
```

Arguments

`file_tbl`	Tibble returned by [import_tracks()], containing at least the columns 'basename' and 'track'.
`track_dir`	Directory containing the track files on disk.
`manifest`	Optional data frame with a 'file' column that matches 'file_tbl$basename'.
`manifest_cols`	Optional named character vector (or list) mapping new column names in 'file_tbl' to column names present in 'manifest'. When 'NULL', all columns aside from 'file' are carried over with the same names.
`mapping`	Optional explicit column mapping passed to [read_tracks()].
`angle_unit`, `time_type`, `tz`, `fps`	Passed to [read_tracks()].
`normalize_xy`	Logical; normalise x/y to unit circle. Default `TRUE`.
`dialect`	Optional dialect name; passed to [read_tracks()].
`keep`, `drop`	Column selection vectors passed to [read_tracks()].
`...`	Additional arguments forwarded to [read_tracks()].

Value

A 'Tracks' with metadata columns replicated for each observation.

`load_tracks` *Legacy helper to merge manifest metadata with a track table*

Description

Legacy helper to merge manifest metadata with a track table

Usage

```
load_tracks(file_tbl, df, track_dir)
```

Arguments

`file_tbl`	Tibble returned by [import_tracks()].
`df`	Data frame containing at least a 'file' column and, optionally, 'arc', 'type', 'obstacle', and 'id'.
`track_dir`	Directory containing the track files (used for validation).

Value

'file_tbl' with additional metadata columns bound in.

`load_tracks2` *Flexible metadata join for track tables*

Description

Flexible metadata join for track tables

Usage

```
load_tracks2(file_tbl, df, track_dir, colnames)
```

Arguments

`file_tbl`	Tibble returned by [import_tracks()].
`df`	Manifest data frame containing a 'file' column.
`track_dir`	Directory containing the track files (used for validation).
`colnames`	Named character vector (or list) mapping the desired column names in the output to columns in 'df'.

Value

'file_tbl' with additional columns appended.

`new_headings_frame` *Low-level headings_frame constructor*

Description

Wraps a data frame as a 'headings_frame' (a tibble subclass) carrying the canonical display/heading metadata. Most users call [headings_frame()] (which validates and normalises angles) or get one from [derive_headings()].

Usage

```
new_headings_frame(
  data,
  display = circ_display(),
  heading_col = "heading",
  colour_col = NULL,
  coords = "absolute",
  color_col = NULL
)
```

Arguments

`data`	A data frame / tibble with the heading column already in unit-circle radians.
`display`	A [circ_display()] object (orientation convention).
`heading_col`	Name of the heading column. Default '"heading"'.
`colour_col`, `color_col`	Optional grouping/colour column name, or 'NULL'. 'color_col' is the American-spelling alias.
`coords`	'"absolute"' or '"relative"'.

Value

A 'headings_frame'.

---

path_straightness *Path straightness index for a single trajectory*

Description

The straightness index is the net (start-to-end) displacement divided by the total path length travelled. It ranges from 0 (a maximally convoluted path that returns to its starting point) to 1 (a perfectly straight path), and is the reciprocal of the tortuosity ratio ([path_tortuosity()]). Unlike that ratio it is bounded and does not blow up when the net displacement is small.

Usage

```
path_straightness(x, y)
```

Arguments

`x`, `y`	Numeric vectors of ordered (in time) coordinates for one trajectory.

Details

The index is scale-invariant: multiplying all coordinates by a constant leaves it unchanged, so absolute or relative coordinates give the same value (provided the transformation is a similarity, i.e. uniform in x and y).

Value

A single straightness value in '[0, 1]', or 'NA_real_' when fewer than two finite points are available or the path has zero length.

See Also

[straightness_index()] for a whole 'Tracks'; [path_tortuosity()].

Examples

```
path_straightness(x = c(0, 1, 2), y = c(0, 0, 0))   # straight -> 1
path_straightness(x = c(0, 1, 0), y = c(0, 1, 0))   # returns to start -> 0
```

`path_tortuosity` *Tortuosity ratio for a single trajectory*

Description

The (classic) tortuosity ratio is the total path length travelled divided by the net (start-to-end) displacement – the reciprocal of the straightness index ([path_straightness()]). It ranges from 1 (a perfectly straight path) upward; larger values indicate a more convoluted path.

Usage

```
path_tortuosity(x, y)
```

Arguments

`x`, `y`	Numeric vectors of ordered (in time) coordinates for one trajectory.

Details

The ratio is unbounded: when a trajectory returns to its starting point the net displacement is zero and the ratio is 'Inf'. For a bounded alternative, or when start and end points may coincide, use [path_straightness()].

Like the straightness index it is scale-invariant.

Value

A single tortuosity value '>= 1', 'Inf' when the net displacement is zero, or 'NA_real_' when fewer than two finite points are available or the path has zero length.

See Also

[tortuosity_ratio()] for a whole 'Tracks'; [path_straightness()].

Examples

```
path_tortuosity(x = c(0, 1, 2), y = c(0, 0, 0))     # straight -> 1
path_tortuosity(x = c(0, 0, 1), y = c(0, 1, 1))     # L-shaped -> sqrt(2)
```

`plot_profile` *Kinematics profile plot for a Tracks*

Description

Draws a per-observation kinematics metric against elapsed time, one line per trajectory – the non-circular companion to [radiate()]. ‘metric = ”speed”‘ plots [instantaneous_speed()]; ‘metric = ”turning”‘ plots [angular_velocity()].

Usage

```
plot_profile(
  ts,
  metric = c("speed", "turning"),
  units = c("radians", "degrees"),
  colour_by = NULL,
  panel_by = NULL
)
```

Arguments

`ts`	A ‘Tracks‘.
`metric`	‘”speed”‘ (default) or ‘”turning”‘.
`units`	For ‘metric = ”turning”‘, ‘”radians”‘ (default) or ‘”degrees”‘ per second.
`colour_by`, `panel_by`	Optional column names of ‘as.data.frame(ts)‘ to colour the lines by / facet into panels. Default: one neutral line per track.

Details

Speed and turning rate are per-second, so a frame rate is required for frame-indexed time ([set_frame_rate()]); POSIXct time is used directly. With a distance calibration ([set_distance_scale()]) speed is in physical units.

Value

A ‘ggplot2‘ object.

See Also

[instantaneous_speed()], [angular_velocity()], [elapsed_seconds()], [radiate()], [set_frame_rate()]

Examples

```
ts <- set_frame_rate(cpunctatus, 30)
plot_profile(ts, metric = "speed")
```

`pose_to_headings` *Derive per-frame headings from pose data without a Tracks*

Description

Computes a heading angle for every row in `df` from either two bodypart keypoint columns or a pre-computed orientation angle column. Intended for tethered or mostly stationary subjects where the position trajectory is absent or uninformative and body pose is the primary signal, but also useful for extracting a dense heading time series from trajectory data. The output is compatible with `circ_dispersion`, `sector_summary`, `add_heading_points`, and `add_angle_rose`.

Usage

```
pose_to_headings(
  df,
  anterior = NULL,
  posterior = NULL,
  theta_col = NULL,
  id_col = NULL,
  time_col = NULL,
  angle_convention = c("unit_circle", "clock")
)
```

Arguments

`df`	Data frame with at least id, time, and keypoint or angle columns.
`anterior`	Prefix of the anterior bodypart columns (`<anterior>_x` and `<anterior>_y` must exist).
`posterior`	Prefix of the posterior bodypart columns.
`theta_col`	Name of a pre-computed orientation angle column (alternative to `anterior`/`posterior`).
`id_col`	Column identifying trials or individuals. Auto-detected from common names; defaults to a single group `"1"` if absent.
`time_col`	Column of frame indices or timestamps. Defaults to row position if absent.
`angle_convention`	`"unit_circle"` (default) or `"clock"`.

Value

Data frame with columns `id`, `time`, `heading` (in radians), with an `angle_convention` attribute for downstream compatibility.

rad_shepherd *Wrap angles to the interval (-pi, pi]*

Description

Wrap angles to the interval (-pi, pi]

Usage

```
rad_shepherd(theta)
```

Arguments

theta	Numeric vector of angles (radians).

Value

Angles wrapped to (-pi, pi].

Examples

```
theta <- seq(from = -5, to = 5, length.out = 6)
rad_shepherd(theta)
```

radial_theme *Themes for radial track plots, named for the ggplot2 base themes.*

Description

Applies one of the standard ggplot2 themes as the look of a ‘radiate()‘ plot, so the panel background, grid lines, and border match the familiar ‘ggplot2::theme_*()‘ appearance. The Cartesian axis text and ticks are not meaningful on a unit-circle plot and are removed by ‘radiate()‘ itself, so this wrapper keeps only the panel-level styling that distinguishes the themes from one another.

Usage

```
radial_theme(name = "void", base_size = 11)
```

Arguments

name	One of ‘"void"‘, ‘"minimal"‘, ‘"classic"‘, ‘"bw"‘, ‘"grey"‘ (or ‘"gray"‘), ‘"light"‘, ‘"dark"‘, ‘"linedraw"‘ – corresponding to the matching ‘ggplot2::theme_*()‘. Default ‘"void"‘.
base_size	Base font size, passed to the underlying theme. Default ‘11‘.

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot() + coord_fixed() + add_circ() + radial_theme("bw")
```

`radiate` *Make ggplot object of tracks radiating from circle centre.*

Description

Accepts either a precomputed data frame of polar/cartesian coordinates or a 'Tracks'. When a 'Tracks' is supplied, column mappings are inferred from the object and handed off to the plotting helpers.

Usage

```
radiate(data, ...)

## S3 method for class 'Tracks'
radiate(data, ...)

## Default S3 method:
radiate(
  data,
  x_col = "rel_x",
  y_col = "rel_y",
  geom = "path",
  group_col = NULL,
  colour_col = NULL,
  colour_cycle = NULL,
  track_colour = c("trajectory", "sequence", "time", "speed"),
  time_units = c("seconds", "minutes", "milliseconds"),
  panel_by = NULL,
  ncol = NULL,
  strip_labels = NULL,
  strip_position = c("top", "bottom", "left", "right", "inside"),
  strip_label_size = 11,
  ticks = NULL,
  degrees = NULL,
  legend = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  axes = NULL,
  angle_labels = c("degrees", "none", "radians"),
  theme = c("void", "minimal", "classic", "bw", "grey", "gray", "light", "dark",
    "linedraw"),
  quadrants = FALSE,
  rings = FALSE,
  grid = c("radial", "cartesian", "none"),
  grid_colour = NULL,
  origin = FALSE,
  circumference = TRUE,
  show_labels = NULL,
  label_col = NULL,
  label_size = 3,
  label_padding = 1.08,
  label_use_repel = TRUE,
  show_tracks = TRUE,
  show_arrow = NULL,
  arrow_angle_col = NULL,
  arrow_colour = "black",
  arrow_colour_col = NULL,
  arrow_size = 2,
  display = circ_display(),
  ...,
  color_col = NULL,
  color_cycle = NULL,
  track_color = NULL,
  grid_color = NULL,
  arrow_color = NULL,
  arrow_color_col = NULL
)

## S3 method for class 'headings_frame'
radiate(
  data,
  col = NULL,
  step = 0.025,
  tol = NULL,
  direction = "inward",
  base_r = 1,
  shade = FALSE,
  shape = FALSE,
  panel_by = NULL,
  ncol = NULL,
  ticks = TRUE,
  degrees = TRUE,
  angle_labels = c("degrees", "none", "radians"),
  title = NULL,
  theme = c("void", "minimal", "classic", "bw", "grey", "gray", "light", "dark",
```

```
    "linedraw"),
  quadrants = FALSE,
  rings = FALSE,
  grid = c("radial", "cartesian", "none"),
  grid_colour = NULL,
  circumference = TRUE,
  origin = FALSE,
  colour_col = NULL,
  legend = FALSE,
  display = circ_display(),
  show_markers = TRUE,
  ...
)
```

Arguments

`data`	Data frame or ‘Tracks‘.
`...`	Additional arguments forwarded to [draw_tracks()].
`x_col`	Name of the x-coordinate column. Default `"rel_x"`.
`y_col`	Name of the y-coordinate column. Default `"rel_y"`.
`geom`	Geom specification passed to [draw_tracks()].
`group_col`	Optional column for grouping aesthetics.
`colour_col`, `color_col`	Optional column for colour aesthetics. Mutually exclusive with ‘colour_cycle‘. ‘color_col‘ is the American-spelling alias.
`colour_cycle`, `color_cycle`	Optional cycling colour specification. Either a positive integer ‘n‘ (trajectories are assigned colours 1–n, cycling back to 1 after every ‘n‘ trajectories) or a character vector of colour values (e.g. ‘c(”red”,”blue”,”green”)‘). When ‘panel_by‘ is set the cycle restarts independently within each panel. Mutually exclusive with ‘colour_col‘. ‘color_cycle‘ is the American-spelling alias.
`track_colour`, `track_color`	How trajectory paths are coloured. ‘”trajectory”‘ (default) keeps the existing per-track colouring. ‘”sequence”‘ colours each path by its point’s normalized position from start (0) to finish (1) within the track, applying a continuous viridis scale with a ‘”start -> finish”‘ colourbar. ‘”time”‘ colours each path by real elapsed time from its own track’s start (see [elapsed_seconds()]), applying a continuous viridis scale with an ‘”elapsed (s)”‘ colourbar; numeric (frame) time requires a [frame_rate()] (set one with ‘set_frame_rate(ts, fps)‘). ‘”speed”‘ colours each path by its instantaneous speed (see [instantaneous_speed()]), applying a continuous viridis scale with a ‘”speed (units/s)”‘ colourbar; numeric (frame) time likewise requires a [frame_rate()]. ‘”sequence”‘, ‘”time”‘, and ‘”speed”‘ own the colour aesthetic and so cannot be combined with ‘colour_col‘/‘colour_cycle‘; overlays render in a fixed colour. The per-track order is taken from the ‘Tracks‘ time column, falling back to row order (with a message) when

no usable time column is present. ‘track_color‘ is the American-spelling alias.

`time_units` Units for ‘track_colour = ”time”‘: ‘”seconds”‘ (default), ‘”minutes”‘, or ‘”milliseconds”‘. Sets the colourbar title and scale.

`panel_by` NULL, a column name, or a character vector of column names to facet by (via [ggplot2::facet_wrap()]). The named column(s) must be present in the data.

`ncol` Number of columns passed to [ggplot2::facet_wrap()] when ‘panel_by‘ is set.

`strip_labels` Logical or ‘NULL‘. Whether to show a label identifying the panel variable value on each panel. Defaults to ‘TRUE‘ when ‘panel_by‘ is set, ‘FALSE‘ otherwise. Ignored when ‘panel_by‘ is ‘NULL‘.

`strip_position`
Position of the panel label. One of ‘”top”‘ (default), ‘”bottom”‘, ‘”left”‘, ‘”right”‘ (ggplot2 strip positions), or ‘”inside”‘ (places a text annotation inside the plot area, centred below the unit circle at y = -1.25).

`strip_label_size`
Font size for strip labels. Applies to both strip text and the in-panel ‘”inside”‘ annotation.

`ticks`, `degrees`, `legend`, `title`, `xlab`, `ylab`, `axes`
Additional styling options. ‘degrees‘ is retained for back-compatibility; ‘degrees = FALSE‘ is equivalent to ‘angle_labels = ”none”‘. Tick styling (colour, width, length) follows the chosen ‘theme‘’s ‘axis.ticks‘.

`angle_labels` One of ‘”degrees”‘ (default; e.g. ‘45°‘), ‘”none”‘, or ‘”radians”‘ (e.g. ‘ /4‘) – the diagonal angle labels around the circle. Label styling (colour, size, family) follows the chosen ‘theme‘’s ‘axis.text‘.

`theme` Plot appearance, named for the ggplot2 base themes: one of ‘”void”‘ (default), ‘”minimal”‘, ‘”classic”‘, ‘”bw”‘, ‘”grey”‘, ‘”light”‘, ‘”dark”‘, or ‘”linedraw”‘. See [radial_theme()].

`quadrants` Logical; draw the two dashed lines through the origin that demarcate the quadrants. Default ‘FALSE‘. Their colour and width follow the chosen ‘theme‘’s grid lines (see [radial_theme()]).

`rings` Logical; draw concentric guide rings (the radial analogue of a grid). Default ‘FALSE‘. Their colour and width follow the chosen ‘theme‘’s grid lines.

`grid` One of ‘”radial”‘ (default), ‘”cartesian”‘, or ‘”none”‘. ‘”radial”‘ replaces the theme’s Cartesian grid with theme-styled radial guides (circular disc + major/minor crosshairs and rings) for grid-bearing themes, and draws nothing for grid-less themes (‘void‘, ‘classic‘). ‘”cartesian”‘ keeps the theme’s square grid; ‘”none”‘ removes all gridlines.

`grid_colour`, `grid_color`
Optional colour overriding the theme-derived guide colour. ‘grid_color‘ is the American-spelling alias.

`origin` Logical; draw a centre point. Default ‘FALSE‘. When drawn it takes the theme’s axis ink colour.

`circumference` Logical; draw the unit-circle circumference as a fallback boundary when no radial grid marks it (grid-less themes, ‘grid = "none"‘). Default ‘TRUE‘. Styled from the theme's ‘axis.line‘ (else the axis ink). On grid-bearing themes the grid's outer ring marks the boundary instead, so this has no effect there.

`show_labels` Whether to place labels at the perimeter.

`label_col` Column containing label values.

`label_size` Text size for perimeter labels.

`label_padding` Multiplier applied to the unit circle when placing labels.

`label_use_repel`
Use ‘ggrepel::geom_text_repel()‘ when available.

`show_tracks` Whether to draw the trajectory paths. Default ‘TRUE‘. Set to ‘FALSE‘ to render the unit circle and any overlays (arrow, circle, ticks) without the track geometry.

`show_arrow` Whether to draw a mean resultant arrow from the centre.

`arrow_angle_col`
Column containing angles (radians) to summarise for the arrow.

`arrow_colour`, `arrow_color`
Arrow colour (a single fixed colour). Ignored when ‘arrow_colour_col‘ is set. ‘arrow_color‘ is the American-spelling alias.

`arrow_colour_col`, `arrow_color_col`
Optional grouping column. When supplied, one mean resultant arrow is drawn per level of this column (within each panel, if ‘panel_by‘ is also set) and mapped to the colour aesthetic, so the arrow can follow a colour grouping independently of faceting. Default ‘NULL‘ draws a single arrow in ‘arrow_colour‘. ‘arrow_color_col‘ is the American-spelling alias.

`arrow_size` Arrow linewidth.

`display` A [‘circ_display‘] object controlling how angles are rendered. Default ‘circ_display()‘ puts North at top with clockwise-positive degrees. Use ‘circ_display(zero = 0)‘ when the reference direction lies at East in unit-circle coordinates (e.g. the ‘cpunctatus‘ dataset).

`col` Name of the angle column in `data`. Defaults to the `heading_col` attribute when `data` is a `headings_frame`.

`step`, `tol`, `direction`, `base_r`, `shade`, `shape`
Passed to `add_stacked_headings`. See that function for details.

`show_markers` When `TRUE` (default) the stacked-dot markers are drawn; `FALSE` returns the themed radial frame only, for callers that layer their own marker and statistic overlays.

Value

A ‘ggplot2‘ object.

American spellings

Every ‘colour...‘ argument and the ‘assign_colour_*‘ / ‘cycle_colours‘ / ‘hf_colour_col‘ functions accept the American ‘color...‘ spelling as an alias (e.g. ‘color‘, ‘color_col‘, ‘track_color‘). British spelling is canonical; supplying both spellings of a pair is an error.

Examples

```
tracks_demo <- simulate_tracks(conditions = data.frame(n_trials = 1L),
                               n_points = 200, seed = 1)
radiate(tracks_demo, x_col = "rel_x", y_col = "rel_y", group_col = "trial_id")
```

---

read_tracks *Construct a Tracks from a data.frame or file(s)*

Description

Construct a Tracks from a data.frame or file(s)

Usage

```
read_tracks(
  x,
  mapping = list(id = NULL, time = NULL, x = NULL, y = NULL, angle = NULL, weight = NULL),
  angle_unit = c("radians", "degrees", "auto"),
  time_type = c("auto", "posix", "seconds", "frames"),
  tz = "UTC",
  fps = NULL,
  normalize_xy = TRUE,
  dialect = NULL,
  dialect_args = list(),
  read_opts = list(delim = NULL, decimal = NULL, sheet = NULL),
  mutate = NULL,
  keep = NULL,
  drop = NULL,
  id_from_filename = TRUE,
  validate = TRUE,
  format = NULL
)
```

Arguments

x	data.frame, file path, or character vector of file paths
mapping	named list for column mapping: id, time, x, y, angle, weight. Any missing will be guessed when possible.
angle_unit	”radians”,”degrees”, or ”auto” to guess from values
time_type	one of ”auto”,”posix”,”seconds”,”frames”

tz	timezone for POSIX times (default "UTC")
fps	frames-per-second when time_type = "frames"
normalize_xy	TRUE to normalize (x,y) to unit circle when both provided
dialect	optional registered dialect name to pre-process raw input
dialect_args	named list of extra arguments forwarded to the dialect function (e.g. `list(bodypart = c("head","thorax"))`)
read_opts	list of file-reading overrides: 'delim' (field separator), 'decimal' (decimal mark), and 'sheet' (Excel worksheet name or number). Any 'NULL' element is auto-detected. Defaults to all-auto.
mutate	list of transformations applied after reading (named functions or formulas)
keep	only keep these columns (NULL = keep all)
drop	drop these columns after mapping
id_from_filename	if TRUE and id missing, derive id from file stem when reading multiple files
validate	if TRUE run S4 validity checks
format	Optional loader format name or list spec registered via [register_loader_format()]

Value

Tracks

`read_tracks_dir` *Read all matching files from a directory and bind into a Tracks*

Description

Read all matching files from a directory and bind into a Tracks

Usage

```
read_tracks_dir(
  dir,
  pattern = "\\.(csv|tsv|txt|parquet|feather)$",
  recursive = FALSE,
  ...
)
```

Arguments

dir	Directory to scan for files
pattern	Regex passed to 'list.files()' to select files
recursive	Recurse into subdirectories when TRUE
...	Additional arguments passed to 'read_tracks()'

`read_tracks_format` *Construct a Tracks from a *format* spec (registered name or inline list)*

Description

Construct a Tracks from a *format* spec (registered name or inline list)

Usage

```
read_tracks_format(x, format, ...)
```

Arguments

`x`	data.frame or path(s)
`format`	registered format name or list spec
`...`	extra args override spec fields

`reference` *Per-trajectory reference direction of a Tracks*

Description

The reference direction (unit-circle radians) against which each trajectory's relative frame (‘rel_theta‘/‘rel_x‘/‘rel_y‘) is defined. Trajectories with no recorded reference default to ‘0‘ (relative frame equals absolute frame).

Usage

```
reference(x)

## S4 method for signature 'Tracks'
reference(x)
```

Arguments

`x`	A [‘Tracks‘].

Value

For ‘reference()‘, a data frame with ‘id‘ and ‘ref_theta‘ (or ‘NULL‘ when none is set).

See Also

[set_reference()], [derive_coords()]

`register_heading_rule` *Register a custom heading derivation rule*

Description

Adds a named function to the heading rule registry so it can be called by `derive_headings` via `rule = "name"`. The function must accept `(df, cols, ...)` and return a data frame with columns `id`, `time`, and `heading` (radians).

Usage

```
register_heading_rule(name, fun, overwrite = FALSE)
```

Arguments

`name`	Character; unique rule name.
`fun`	Function with signature `function(df, cols, ...)`.
`overwrite`	Logical; replace an existing rule of the same name.

Value

The rule name, invisibly.

See Also

`list_heading_rules`, `derive_headings`

`register_loader_dialect` *Register a custom loader dialect The function must accept (x, ...) and return a data.frame in long form with columns at least id,time and one of (angle) or (x,y)*

Description

Register a custom loader dialect The function must accept (x, ...) and return a data.frame in long form with columns at least id,time and one of (angle) or (x,y)

Usage

```
register_loader_dialect(name, fun, overwrite = FALSE)
```

Arguments

`name`	Unique dialect name
`fun`	Function that accepts (x, ...) and returns a data.frame with id/time/angle or id/time/x/y
`overwrite`	Replace an existing dialect registered with the same name

---

`register_loader_format` *Register a declarative loader *format* (list or YAML/JSON file) The spec maps cleanly onto read_tracks() args and supports regex-based column finding.*

Description

Register a declarative loader *format* (list or YAML/JSON file) The spec maps cleanly onto read_tracks() args and supports regex-based column finding.

Usage

```
register_loader_format(name, spec, overwrite = FALSE)
```

Arguments

`name`	A unique name
`spec`	A named list, or a path to a YAML/JSON file defining the spec
`overwrite`	Overwrite an existing format of the same name

`sector_summary` *Proportion of time spent in angular sectors*

Description

Bins heading angles into sectors and returns count and proportion per sector, optionally grouped by trial or condition. Useful for dwell-time analysis of dense per-frame heading series (e.g. gaze direction from a tethered subject).

Usage

```
sector_summary(hd, sectors = 8L, group_col = NULL, angle_col = "heading")
```

Arguments

`hd`	Data frame containing headings in radians.
`sectors`	Either a single integer (number of equal sectors spanning the full circle, default `8`) or a numeric vector of break points in radians. Break points need not include $\pm\pi$; they are added automatically.
`group_col`	Column(s) to group by. `NULL` uses all rows.
`angle_col`	Name of the heading column. Default `"heading"`.

Value

Data frame with columns `group_col` (if supplied), `sector` (degree label), `mid_angle` (sector midpoint in radians), `count`, `proportion`.

`set_reference` *Set the per-trajectory reference and re-derive the relative frame*

Description

Updates each trajectory's reference direction and re-derives its relative columns (‘rel_theta‘/‘rel_x‘/‘rel_y‘) from the unit-circle position via [derive_coords()], so the relative frame stays consistent. The step is recorded in [transform_history()]. This is the drift-safe way to change the reference frame – prefer it over a manual [apply_transform()].

Usage

```
set_reference(x, value)

## S4 method for signature 'Tracks'
set_reference(x, value)
```

Arguments

`x`	A [‘Tracks‘] with position roles (‘cols$x‘/‘cols$y‘) and relative roles (‘cols$angle‘/‘cols$rel_x‘/‘cols registered.
`value`	Reference direction(s) in unit-circle radians: a scalar applied to all trajectories, or a named numeric vector / two-column (‘id‘,‘ref_theta‘) data frame setting them per trajectory.

Value

The updated [‘Tracks‘].

See Also

[reference()], [derive_coords()]

`simulate_tracks` *Simulate trajectory sets under configurable experimental conditions*

Description

This module generates synthetic trajectories whose directional concentration and tortuosity vary across experimental conditions and continuous predictors. It is useful for creating reproducible examples, tutorials, and test fixtures without relying on large raw tracking files.

Usage

```
simulate_tracks(
  n_points = 200,
  conditions = NULL,
  output = c("tibble", "trajset", "both"),
  write_path = NULL,
  seed = NULL,
  radial_noise = 0.02,
  phi = 0.85,
  frame_rate = NULL
)
```

Arguments

`n_points`	Integer number of frames per trajectory.
`conditions`	Optional data frame describing experimental conditions. See Details for column descriptions. When omitted, a three-condition template is used.
`output`	Character string controlling the return type: "tibble" (default) returns a long-form data frame, "trajset" returns a [Tracks] object, and "both" returns a list containing both representations.
`write_path`	Optional file path (CSV) to which the simulated data should be written.
`seed`	Optional integer seed supplied to [set.seed()] for reproducibility.
`radial_noise`	Standard deviation of radial noise applied to the unit radius profile.
`phi`	Autocorrelation parameter (0-1) used when generating angular noise series; higher values produce smoother paths.
`frame_rate`	Optional capture frame rate (frames per second) attached to the returned [Tracks] via [set_frame_rate()] when ‘output‘ is ‘"trajset"‘ or ‘"both"‘. The default ‘NULL‘ leaves the output unchanged.

Details

The ‘conditions‘ data frame can contain the following columns (defaults are supplied when missing):

- ‘condition‘ (character): condition label. - ‘n_trials‘ (integer): number of trajectories to simulate for the condition. - ‘ref_mean‘ (numeric radian): baseline reference heading (default 0). - ‘mean_slope‘ (numeric, default 0): per-condition slope shifting the mean heading with the predictor. The effective per-trial mean is ‘ref_mean + mean_slope * predictor‘ and is recorded in ‘ref_heading‘. A default of 0 reproduces the historical seeded output byte-for-byte. - ‘concentration_base‘ (numeric): baseline von Mises concentration (kappa). - ‘concentration_slope‘ (numeric): optional slope applied to the predictor. - ‘tortuosity_base‘ (numeric): baseline angular noise scale. - ‘tortuosity_slope‘ (numeric): slope applied to the predictor. - ‘tortuosity_sd‘ (numeric): additional random variation per trial. - ‘predictor_mean‘, ‘predictor_sd‘ (numeric): parameters used to sample the per-trial predictor when explicit values are not supplied. - ‘predictor_values‘ (list-column): optional explicit predictor values (length ‘n_trials‘) overriding the generated values. - ‘modality‘ (character): the sample modality from which per-trial principal headings are drawn. One of ‘”unimodal”‘ (default), ‘”uniform”‘, ‘”axial”‘, or ‘”multimodal”‘. Controls the *distribution of headings across trials*, not the within-trial path shape. - ‘n_modes‘ (integer): number of evenly spaced modes used when ‘modality == ”multimodal”‘ (default 1; ignored by other modalities). - ‘track_shape‘ (character): the *within-track* path shape. One of ‘”directed”‘ (default) – a single sweep towards ‘final_heading‘ – or ‘”oscillatory”‘ – back-and-forth motion along the principal axis ‘final_heading‘. Oscillatory tracks form a genuinely axial position cloud, so the position-based axial methods (‘pca_axis‘, ‘ransac_straight‘) recover the axis at default settings; the directional methods (e.g. ‘net‘) cancel. The step-based ‘velocity_axis‘ recovers the axis only when sampling is coarse enough that the per-step axial motion exceeds the perpendicular line-width jitter (see the ‘line_width‘ note below). - ‘n_reversals‘ (integer): number of direction reversals in an oscillatory track (default 3; ignored when ‘track_shape == ”directed”‘). - ‘amplitude‘ (numeric): peak excursion along the axis for an oscillatory track, clamped to ‘[1e-3, 1]‘ (default 0.9; ignored when directed). - ‘line_width‘ (numeric): half-width of an oscillatory track expressed as a fraction of ‘amplitude‘, controlling the perpendicular Gaussian jitter (default 0.05, clamped to ‘[1e-4, 1]‘; ignored when directed). The line-width is intentionally independent of ‘tortuosity_*‘, so the track is a genuinely thin line and the principal axis is recoverable by the position-based methods (‘pca_axis‘, ‘ransac_straight‘) at default settings. The step-based ‘velocity_axis‘ is sampling-density sensitive: because the per-frame along-axis step shrinks with ‘n_points‘ while the perpendicular jitter step does not, dense sampling lets the jitter dominate and the estimate flips toward the perpendicular.

The predictor can represent any continuous covariate (e.g. reference intensity). The final heading concentration increases with larger kappa, whereas larger tortuosity values produce more sinuous paths.

For each trial the principal (final) heading is drawn according to the condition’s ‘modality‘: ‘”unimodal”‘ draws from a single von Mises about ‘ref_mean‘; ‘”uniform”‘ draws from a circular uniform distribution; ‘”axial”‘ draws from a von Mises about ‘ref_mean‘ or ‘ref_mean + pi‘ with equal probability; ‘”multimodal”‘ draws from one of ‘n_modes‘ von Mises components evenly spaced around the circle starting at ‘ref_mean‘. The ‘”unimodal”‘ branch is identical to the historical draw, so seeded output is unchanged from earlier versions.

For an ‘”oscillatory”‘ track the position sweeps back and forth along the axis ‘final_heading‘ following a deterministic triangle wave of amplitude ‘amplitude‘ with ‘n_reversals‘ direction changes, plus a small Gaussian jitter perpendicular to the axis with standard deviation ‘amplitude * line_width‘. This line-width is independent of the tortuosity settings, so the track stays a thin line and the axis remains recoverable by the position-based methods

(‘pca_axis‘, ‘ransac_straight‘) at default settings. The step-based ‘velocity_axis‘ recovers the same axis only at coarse sampling, since its per-step axial signal shrinks with ‘n_points‘ while the perpendicular jitter step does not. The ‘"directed"‘ branch is byte-identical to the historical geometry (and never draws the perpendicular jitter), so the default seeded output is unchanged.

Every simulated row records the ground-truth generating structure in additional columns: ‘modality‘ (character), ‘n_modes‘ (integer), ‘mode_id‘ (integer index of the component the heading came from), ‘mode_mean‘ (numeric radian mean of that component as the un-wrapped generating centre, e.g. ‘ref_mean + pi‘ for the second axial pole, versus ‘final_heading‘ which is the wrapped draw; ‘NA‘ for ‘"uniform"‘), ‘track_shape‘ (character), ‘n_reversals‘ (integer), ‘amplitude‘ (numeric) and ‘line_width‘ (numeric). When a ‘Tracks‘ is returned, the resolved generating conditions are stored in ‘meta$sim_conditions‘.

Value

Depending on ‘output‘, a tibble, a ‘Tracks‘, or a list containing both. When ‘write_path‘ is supplied the data are also written to disk.

See Also

circ_model_select, test_uniformity, derive_headings

Examples

```
sim <- simulate_tracks(seed = 1)
head(sim)

# Request both tibble and Tracks representations
sim_both <- simulate_tracks(output = "both", seed = 123)
names(sim_both)

example_conditions <- tibble::tibble(
  condition = paste0("level_", 1:4),
  n_trials = 30L,
  ref_mean = seq(-pi/8, pi/8, length.out = 4),
  concentration_base = c(2, 4, 6, 8),
  concentration_slope = 0.8,
  tortuosity_base = c(0.12, 0.09, 0.06, 0.04),
  tortuosity_slope = -0.01,
  tortuosity_sd = 0.015,
  predictor_mean = seq(-1, 1, length.out = 4),
  predictor_sd = 0.15
)
sim_four_levels <- simulate_tracks(
  n_points = 120,
  conditions = example_conditions,
  output = "both",
  seed = 42
)
aggregate(trial_id ~ condition, data = sim_four_levels$tibble, length)
names(sim_four_levels)
```

`stack_headings` *Add stacking columns to a headings data frame*

Description

Computes radial positions for stacked dot plots on circular plots. Observations at the same angle (or within `tol` radians of each other) are assigned successive radial positions, preventing overplotting of coincident or binned headings.

Usage

```
stack_headings(
  data,
  col = NULL,
  step = 0.025,
  start_sep = 0,
  tol = NULL,
  direction = "inward",
  base_r = 1,
  shade = FALSE,
  shape = FALSE,
  group = NULL
)
```

Arguments

`data`	A data frame with an angle column in radians.
`col`	Name of the angle column. Defaults to the `heading_col` attribute when `data` is a `headings_frame`.
`step`	Radial gap between successive dots in a stack, in data units (the analogue of `circular::plot.circular`'s `sep`). Default `0.025` matches that package; larger values separate the dots more.
`start_sep`	Radial offset of the first (outermost, for `"inward"`) dot from `base_r`, in data units (the analogue of `circular::plot.circular`'s `start.sep`). Default `0` places the first dot on the reference circle. A small positive value shifts the whole stack off the line so the dots abut rather than straddle it.
`tol`	Grouping tolerance in radians. `NULL` (default) = exact equality, correct for binned data. `tol > 0` assigns each observation to the nearest group centre within `tol` radians (greedy, sorted-order scan); angles near `0` and `2*pi` are not treated as neighbours.
`direction`	`"inward"` (default, stacks toward centre) or `"outward"` (away from perimeter, matches `circular` default).
`base_r`	Radius of the reference circle in data units. Default 1.

`shade`	If `TRUE`, add a `shade_n` column (alias of `stack_n`) for use as an alpha aesthetic.
`shape`	If `TRUE`, add a `shape_code` integer column: 1 = hollow (outermost / singleton), 2 = filled (middle), 3 = filled with ring (innermost in a stack of 3+).
`group`	Optional column name; when set, stacking is computed independently within each group and the rows recombined. Default `NULL`.

Value

`data` augmented with `stack_r` and `stack_n` columns (always), plus `shade_n` and/or `shape_code` when requested. Row count is unchanged.

See Also

`headings_frame`, `add_stacked_headings`

`step_speed` *Per-step speed along a trajectory*

Description

Speed of each step as straight-line step distance divided by the elapsed time of that step: ‘sqrt(diff(x)^2 + diff(y)^2) / diff(seconds)‘. The unit is the distance unit of ‘x‘/‘y‘ per second; for radiatR’s unit-arena coordinates that is arena-units (radii) per second.

Usage

```
step_speed(x, y, seconds)
```

Arguments

`x`, `y`	Numeric vectors of ordered (in time) coordinates for one trajectory.
`seconds`	Numeric vector, the elapsed time of each point in seconds (same length as ‘x‘/‘y‘).

Value

A numeric vector of per-step speeds, length ‘length(x) - 1‘. A step is ‘NA‘ when either endpoint is non-finite or its time increment is ‘<= 0‘; ‘numeric(0)‘ when fewer than two points are given.

See Also

[track_speed()] for a whole ‘Tracks‘; [elapsed_seconds()].

Examples

```
step_speed(x = 0:3, y = rep(0, 4), seconds = (0:3) / 30)   # 30 units/s
```

straightness_index *Per-trajectory straightness index for a Tracks*

Description

Computes [path_straightness()] for each trajectory in a ‘Tracks‘, ordering each trajectory’s points by its time column when one is recorded.

Usage

```
straightness_index(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

`ts`	A ‘Tracks‘.
`x_col`, `y_col`	Names of the coordinate columns to use. Default to the ‘Tracks‘’s recorded x/y columns (the real recorded positions), so the metric reflects the physical path rather than any display transform.

Value

A ‘data.frame‘ with one row per trajectory: the ‘Tracks‘’s id column and a numeric ‘straightness‘ column.

See Also

[path_straightness()], [tortuosity_ratio()]

test_concentration *Test whether groups share the same concentration (dispersion)*

Description

Two tests are available:

`parametric = TRUE` **(default)** Likelihood-ratio test for equal von Mises κ across groups (`equal.kappa.test`). Returns a chi-squared statistic with $k - 1$ degrees of freedom.

`parametric = FALSE` Wallraff’s non-parametric test for equal angular dispersions – no distributional assumption required.

Usage

```
test_concentration(
  hd,
  group_col,
  angle_col = "heading",
  parametric = TRUE,
  axial = FALSE
)
```

Arguments

`hd`	Data frame with heading and group columns.
`group_col`	Column identifying conditions or groups.
`angle_col`	Heading column in radians. Default `"heading"`.
`parametric`	Logical. `TRUE` (default) uses `equal.kappa.test`; `FALSE` uses `wallraff.test`.
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: the test is run via the angle-doubling method, comparing axial concentrations. Default 'FALSE' (ordinary directional data).

Value

One-row tidy data frame with `statistic`, `df` (parametric only), `p_value`, and `test`.

`test_mean_directions` *Test whether groups share the same mean direction (Watson-Williams)*

Description

Wraps `watson.williams.test` – the circular analogue of the parametric F-test for equal means. Assumes von Mises- distributed data with equal concentrations across groups; if concentrations differ substantially or the distribution is non-von-Mises, consider a non-parametric alternative.

Usage

```
test_mean_directions(
  hd,
  group_col,
  angle_col = "heading",
  pairwise = FALSE,
  p_adjust = "none",
  axial = FALSE
)
```

Arguments

`hd`	Data frame with heading and group columns.
`group_col`	Column identifying conditions or groups.
`angle_col`	Heading column in radians. Default `"heading"`.
`pairwise`	Logical. `FALSE` (default) returns a single omnibus test across all groups. `TRUE` returns all pairwise comparisons.
`p_adjust`	Multiple-comparison correction method passed to `p.adjust`. Default `"none"`. Applies only to the pairwise output; a `p_value_adj` column is added. Strongly recommended when `pairwise = TRUE`: use `"BH"` (Benjamini-Hochberg) or `"holm"` (family-wise control). Ignored for the omnibus test (single p-value, no adjustment needed).
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: the test is run via the angle-doubling method, comparing group axes. Default 'FALSE' (ordinary directional data).

Value

Tidy data frame. Omnibus result has columns `n_groups`, `statistic`, `df1`, `df2`, `p_value`, `test`. Pairwise result additionally has `group1`, `group2`, and `p_value_adj` (when `p_adjust != "none"`).

`test_uniformity` *Per-group tests of circular uniformity*

Description

Tests whether each group's headings are uniformly distributed (i.e. no preferred direction), using any of four classical tests. The Rayleigh test (`"rayleigh"`, default) returns an exact numeric p-value; the other three (`"kuiper"`, `"rao"`, `"watson"`) use look-up tables and the `p_value` column contains the tabled significance level rather than a continuous p-value.

Usage

```
test_uniformity(
  hd,
  group_col = NULL,
  angle_col = "heading",
  test = c("rayleigh", "kuiper", "rao", "watson", "hermans_rasson"),
  p_adjust = "none",
  axial = FALSE,
  n_sim = 9999L
)
```

Arguments

`hd`	Data frame with a heading column in radians.
`group_col`	Column to group by. `NULL` tests the whole data frame as one group.
`angle_col`	Heading column name. Default `"heading"`.
`test`	One of `"rayleigh"` (default), `"kuiper"`, `"rao"`, `"watson"`, or `"hermans_rasson"`. The last is the Hermans-Rasson omnibus test (Landler, Ruxton & Malkemper 2019), far more powerful than Rayleigh against multimodal / non-symmetric alternatives; its `p_value` is obtained by Monte-Carlo simulation.
`p_adjust`	Multiple-comparison correction method passed to `p.adjust`. Default `"none"`. Applies only when `group_col` is supplied; a `p_value_adj` column is added to the result. Recommended: `"BH"` (Benjamini-Hochberg) when testing many conditions. Only meaningful for the Rayleigh test (exact p-values).
`axial`	Logical. Treat the angles as axial (bidirectional, mod-pi) data: the uniformity test is run via the angle-doubling method (testing for an axis). Default 'FALSE' (ordinary directional data).
`n_sim`	Number of Monte-Carlo replicates for the `"hermans_rasson"` p-value. Default `9999`. Ignored by the other tests. Set the RNG seed with `set.seed` for reproducible p-values.

Value

Tidy data frame with columns `group_col` (if supplied), `statistic`, `p_value`, `n`, `test`, and `p_value_adj` (when `p_adjust != "none"`).

References

Landler, L., Ruxton, G.D. & Malkemper, E.P. (2019). The Hermans-Rasson test as a powerful alternative to the Rayleigh test for circular statistics in biology. BMC Ecology 19:30. doi:10.1186/s1289801902468.

`tortuosity_ratio` *Per-trajectory tortuosity ratio for a Tracks*

Description

Computes [path_tortuosity()] for each trajectory in a 'Tracks', ordering each trajectory's points by its time column when one is recorded.

Usage

```
tortuosity_ratio(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

`ts`	A ‘Tracks‘.
`x_col`, `y_col`	Names of the coordinate columns to use. Default to the ‘Tracks‘’s recorded x/y columns (the real recorded positions), so the metric reflects the physical path rather than any display transform.

Value

A ‘data.frame‘ with one row per trajectory: the ‘Tracks‘’s id column and a numeric ‘tortuosity‘ column (‘>= 1‘, possibly ‘Inf‘).

See Also

[path_tortuosity()], [straightness_index()]

`track_duration` *Duration of each track*

Description

Total elapsed time of each trajectory (its last point minus its first), computed via [elapsed_seconds()].

Usage

```
track_duration(ts, units = c("seconds", "minutes", "milliseconds"))
```

Arguments

`ts`	A [Tracks] object.
`units`	‘"seconds"‘ (default), ‘"minutes"‘, or ‘"milliseconds"‘.

Value

A data frame with columns ‘id‘ and ‘duration‘.

See Also

[frame_rate()], [elapsed_seconds()]

track_length *Per-trajectory path length for a Tracks*

Description

Total distance travelled along each trajectory. With a distance calibration set ([set_distance_scale()]) the result is in physical units; otherwise in the units of the recorded ‘x‘/‘y‘ coordinates.

Usage

```
track_length(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

ts	A ‘Tracks‘.
x_col, y_col	Names of the coordinate columns. Default to the ‘Tracks‘’s recorded x/y columns.

Value

A ‘data.frame‘ with one row per trajectory: the id column and a numeric ‘length‘ column.

See Also

[set_distance_scale()], [track_speed()], [straightness_index()]

track_speed *Per-trajectory speed for a Tracks, in real units*

Description

Summarises each trajectory’s speed (distance per second). Step speeds come from [step_speed()] using the track’s elapsed time from [elapsed_seconds()]; the per-track summary is the chosen ‘stat‘ of those step speeds.

Usage

```
track_speed(
  ts,
  stat = c("mean", "max", "median"),
  x_col = ts@cols$x,
  y_col = ts@cols$y
)
```

Arguments

`ts`	A ‘Tracks‘.
`stat`	Per-track reduction of the step speeds: ‘"mean"‘ (default), ‘"max"‘, or ‘"median"‘.
`x_col`, `y_col`	Names of the coordinate columns. Default to the ‘Tracks‘’s recorded x/y columns.

Details

Numeric (frame) time requires a frame rate ([set_frame_rate()]); POSIXct time is used directly. With the default coordinate columns the unit is arena-units (radii) per second, because radiatR normalises trajectories to a unit arena.

Value

A ‘data.frame‘ with one row per trajectory: the id column and a numeric ‘speed‘ column (‘NA‘ for tracks with fewer than two usable points).

See Also

[step_speed()], [elapsed_seconds()], [track_duration()], [straightness_index()]

`track_turning` *Per-trajectory turning-rate summary for a Tracks*

Description

Summarises each trajectory’s [angular_velocity()] (signed turning rate, counter-clockwise positive) by the chosen statistic.

Usage

```
track_turning(
  ts,
  stat = c("mean_abs", "mean", "max_abs", "median_abs"),
  units = c("radians", "degrees"),
  x_col = ts@cols$x,
  y_col = ts@cols$y
)
```

Arguments

`ts`	A ‘Tracks‘.
`stat`	‘"mean_abs"‘ (default) the typical turning magnitude; ‘"mean"‘ the signed net bias (opposite turns cancel); ‘"max_abs"‘ the sharpest turn; ‘"median_abs"‘ a robust turning magnitude.

`units`	'"radians"' per second (default) or '"degrees"' per second.
`x_col`, `y_col`	Names of the coordinate columns. Default to the 'Tracks''s recorded x/y columns.

Value

A 'data.frame' with one row per trajectory: the id column and a numeric 'turning' column ('NA' for tracks with fewer than three points).

See Also

[angular_velocity()], [track_velocity()], [track_speed()]

`track_velocity` *Per-trajectory net velocity for a Tracks*

Description

The net (average) velocity vector of each trajectory: its straight-line displacement divided by its elapsed duration. The magnitude is the net speed and 'atan2(vy, vx)' the overall direction of travel (so a path returning to its start has zero net velocity). Distance-calibrated when a scale is set.

Usage

```
track_velocity(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

`ts`	A 'Tracks'.
`x_col`, `y_col`	Names of the coordinate columns. Default to the 'Tracks''s recorded x/y columns.

Details

Numeric (frame) time requires a frame rate ([set_frame_rate()]); POSIXct time is used directly.

Value

A 'data.frame' with one row per trajectory: the id column and numeric 'vx', 'vy' ('NA' for tracks with fewer than two points or zero duration).

See Also

[velocity_vector()], [track_speed()], [track_turning()], [set_distance_scale()]

`Tracks-class` *Tracks container for circular trajectories*

Description

Tracks container for circular trajectories

Construct a Tracks

Usage

```
tracks(
  df,
  id = "id",
  time = "time",
  angle = NULL,
  x = NULL,
  y = NULL,
  rel_x = NULL,
  rel_y = NULL,
  angle_unit = NULL,
  weight = NULL,
  normalize_xy = TRUE,
  meta = list(),
  transform_history = NULL
)

## S4 method for signature 'Tracks'
length(x)

## S4 method for signature 'Tracks,ANY,missing,missing'
x[i]

## S4 method for signature 'Tracks'
c(x, ..., recursive = FALSE)
```

Arguments

`df`	data.frame in long form
`id`, `time`	Columns naming trajectory id and time
`angle`	Optional angle column (radians or degrees, see angle_unit)
`x`, `y`	Optional cartesian columns; if provided, converted to unit circle and angle inferred
`rel_x`, `rel_y`	Optional column names for pre-transformed relative coordinates (centred on the origin, unit-circle scaled). Both must be supplied together or not at all.

`angle_unit`	Units of provided angle ("radians" or "degrees"); stored internally as radians
`weight`	Optional weight column name
`normalize_xy`	If TRUE (default), (x,y) are scaled to the unit circle per trajectory: each trajectory is centred on its bounding-box midpoint and scaled so its furthest point sits at radius 1. This preserves trajectory shape and places the origin at the centre (what the radius-based heading rules expect). Raw coordinates are retained in '<x>_raw'/'<y>_raw'. If FALSE, (x,y) are kept as supplied. (Landmark-based mapping, when available, is more accurate; this is the no-landmark fallback.)
`meta`	Free-form list of metadata
`transform_history`	Optional tibble describing transformation steps applied to the trajectories. Must contain columns 'step', 'order', 'id', 'implementation', 'params', and 'depends_on'.
`i`	Trajectory identifiers (character ids, numeric indices, or logical vector)
`...`	Additional Tracks objects to append
`recursive`	Ignored; maintained for signature compatibility

Value

A Tracks S4 object

Slots

`data` data.frame of trajectory observations in long form

`cols` list mapping required column names (id/time/angle/optional x,y,weight)

`angle_unit` character describing the original angle unit supplied

`meta` list of additional metadata attached to the set

`transform_history` *Transform history helpers for Tracks objects*

Description

Transform history helpers for Tracks objects

Usage

```
transform_history(x)

## S4 method for signature 'Tracks'
transform_history(x)

log_transform(
```

```
  x,
  step,
  traj_ids = NULL,
  implementation = step,
  params = NULL,
  order = NULL,
  depends_on = NULL
)

## S4 method for signature 'Tracks'
log_transform(
  x,
  step,
  traj_ids = NULL,
  implementation = step,
  params = NULL,
  order = NULL,
  depends_on = NULL
)

set_transform_history(x, history)

## S4 method for signature 'Tracks'
set_transform_history(x, history)
```

Arguments

x	A ‘Tracks‘ object.
step	Character identifier for the transform step.
traj_ids	Character vector of trajectory identifiers affected by the step. Defaults to all trajectories in ‘x‘ when ‘NULL‘.
implementation	Character label for the implementation used to apply the step. Defaults to ‘step‘.
params	List-column of per-trajectory parameter sets (recycled when a single entry is provided).
order	Optional integer giving the execution order. When omitted the step is appended to the end of the log.
depends_on	Optional character vector naming prerequisite step(s).
history	Tibble or list describing the full transform history to replace.

Value

For ‘transform_history‘, a tibble describing the recorded steps. For ‘log_transform‘ and ‘set_transform_history‘, the updated ‘Tracks‘ object.

`velocity_vector` *Per-observation velocity vector for a Tracks*

Description

The velocity components (‘vx‘, ‘vy‘) at each point, aligned to the ‘Tracks‘’s rows. Each point carries the velocity of the step that ends at it (step displacement divided by its elapsed time); the first point of every trajectory is ‘NA‘. The magnitude is [instantaneous_speed()] and the direction is ‘atan2(vy, vx)‘.

Usage

```
velocity_vector(ts, x_col = ts@cols$x, y_col = ts@cols$y)
```

Arguments

`ts`	A ‘Tracks‘.
`x_col`, `y_col`	Names of the coordinate columns. Default to the ‘Tracks‘’s recorded x/y columns.

Details

Numeric (frame) time requires a frame rate ([set_frame_rate()]); POSIXct time is used directly. With a distance calibration ([set_distance_scale()]) the components are in physical units per second; otherwise coordinate units per second.

Value

A ‘data.frame‘ with columns ‘vx‘ and ‘vy‘, one row per observation in ‘ts@data‘ order (‘NA‘ at each trajectory’s first point).

See Also

[instantaneous_speed()], [angular_velocity()], [set_distance_scale()]

`vonmises_fit` *Fit a von Mises distribution to per-group heading data*

Description

Estimates the mean direction μ and concentration κ of a von Mises distribution via maximum likelihood, together with asymptotic standard errors and a confidence interval on μ. Intended as a parametric companion to `circ_dispersion`: where `circ_dispersion` returns the empirical resultant length R and circular SD, `vonmises_fit` returns the MLE $\hat{\kappa}$ with its uncertainty.

Usage

```
vonmises_fit(
  hd,
  group_col = NULL,
  angle_col = "heading",
  conf = 0.95,
  axial = FALSE
)
```

Arguments

`hd`	Data frame containing headings in radians.
`group_col`	Column(s) to group by. `NULL` fits a single model to all rows.
`angle_col`	Name of the heading column. Default `"heading"`.
`conf`	Confidence level for the interval on μ. Default `0.95`.
`axial`	Logical; when 'TRUE', fit an axial (bidirectional, mod-pi) von Mises via the doubled-angle method: 'mu'/'mu_deg' are the mean **axis** in [0, pi), 'kappa' is the concentration about that axis (estimated in the doubled-angle frame), and 'se_mu'/'ci_lo'/'ci_hi' are halved accordingly. Default 'FALSE' (directional).

Details

$\kappa = 0$ corresponds to a uniform distribution (no preferred direction); larger values indicate increasing concentration. The confidence interval on μ uses a normal approximation and is unreliable for $\kappa < 0.5$ or small samples.

Value

Data frame with columns `group_col` (if supplied), `mu` (MLE mean direction, radians), `mu_deg` (degrees), `kappa` (MLE concentration), `se_mu`, `se_kappa` (asymptotic standard errors), `ci_lo` and `ci_hi` (`conf`-level interval on μ, radians), `n`.

`wrappedcauchy_fit` *Fit a wrapped Cauchy distribution to per-group heading data*

Description

Estimates the mean direction μ and concentration ρ of a wrapped Cauchy distribution via maximum likelihood. The wrapped Cauchy has heavier tails than the von Mises and is more appropriate for data with outliers, weak or noisy directionality, or when a von Mises fit looks visually poor on a rose diagram.

Usage

```
wrappedcauchy_fit(hd, group_col = NULL, angle_col = "heading", axial = FALSE)
```

Arguments

`hd`	Data frame containing headings in radians.
`group_col`	Column(s) to group by. `NULL` fits a single model.
`angle_col`	Name of the heading column. Default `"heading"`.
`axial`	Logical; when 'TRUE', fit an axial (bidirectional, mod-pi) wrapped Cauchy via the doubled-angle method: 'mu'/'mu_deg' are the mean **axis** in [0, pi) and 'rho' is the concentration about that axis (estimated in the doubled-angle frame). Default 'FALSE' (directional).

Details

$\rho = 0$ is a uniform distribution (no preferred direction); $\rho = 1$ is a point mass (perfect concentration). Unlike von Mises κ, the wrapped Cauchy ρ is bounded to $[0, 1)$.

Standard errors are not computed by `mle.wrappedcauchy`; check `convergence` is the `optim` return code (0 = fully converged; 1 = iteration limit reached but estimates are typically still reliable). For uncertainty estimation use `vonmises_fit` with the same data and compare model fits visually via `add_vonmises_density` and `add_wrappedcauchy_density`.

Value

Data frame with columns `group_col` (if supplied), `mu` (MLE mean direction, radians), `mu_deg` (degrees), `rho` (concentration, 0–1), `convergence` (0 = converged), `n`.

See Also

`vonmises_fit`, `add_wrappedcauchy_density`

`zone_dwell` *Dwell-time proportions across quadrant x ring zones*

Description

Classifies each trajectory observation into one of N quadrant sectors and M annular rings, then returns per-trial frame counts and proportions. Applicable to any circular-field analysis where spatial dwell time is of interest (e.g. water maze, open-field, Drosophila preference assay).

Usage

```
zone_dwell(
  x,
  target_angle,
  target_radius = 1,
  ring_breaks = c(0, 0.5, 0.8, 1),
  coords = c("absolute", "relative")
)
```

Arguments

`x`	A [‘Tracks‘] object with x/y (or rel_x/rel_y) columns registered.
`target_angle`	Numeric. Radians. Direction of the target zone from the origin. Q1 spans +/-45degrees around this angle.
`target_radius`	Numeric. Accepted for API symmetry with [count_goal_entries()] but not used in zone assignment. Default ‘1‘.
`ring_breaks`	Numeric vector. Annular ring boundaries, must start at ‘0‘. Default ‘c(0, 0.5, 0.8, 1)‘ gives three rings: inner / middle / outer (thigmotaxis).
`coords`	Character. ‘”absolute”‘ (default) uses ‘@cols$x‘/‘@cols$y‘; ‘”relative”‘ uses ‘@cols$rel_x‘/‘@cols$rel_y‘.

Details

The target quadrant (Q1) is centred on ‘target_angle‘; Q2–Q4 follow counter-clockwise. Observations outside ‘max(ring_breaks)‘ are excluded from both counts and the proportion denominator.

Value

A ‘data.frame‘ with one row per observed (id x quadrant x ring) combination, with columns ‘id‘, ‘quadrant‘ (integer, 1 = target), ‘ring‘ (integer, 1 = innermost), ‘zone‘ (e.g. ‘”Q1.R3”‘), ‘n_frames‘ (integer), and ‘proportion‘ (numeric). Combinations with zero observations are omitted.

See Also

[count_goal_entries()]

Examples

```
## Not run:
# Water maze probe trial: platform was at 45 degrees (NE)
dwell <- zone_dwell(ts, target_angle = pi / 4,
                    ring_breaks = c(0, 0.5, 0.8, 1))
# Q1 proportion > 0.25 indicates above-chance target preference

## End(Not run)
```

Index