

Package: radiatR (via r-universe)

October 27, 2026

Type Package

Title Analyse and Visualise Headings and Trajectories on the Unit Disk

Version 0.1.3

Description A complete pipeline for analysing directional data -- headings and trajectories on the unit disk. Supply a table of angles directly, or reconstruct headings from movement trajectories imported from tracking tools ('EthoVision', 'DeepLabCut', 'SLEAP', 'TRex', 'ANY-maze', 'TrackMate', 'idtracker.ai', 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), circular-linear regression (Fisher and Lee (1992) <[doi:10.2307/2532334](#)>), model selection among orientation models (Fitak and Johnsen (2017) <[doi:10.1242/jeb.167056](#)>), and uniformity and multi-sample hypothesis tests (Rayleigh, Watson-Williams, Hermans-Rasson (Landler, Ruxton and Malkemper (2019) <[doi:10.1186/s12898-019-0246-8](#)>), and others) with multiple-comparison correction. Path tortuosity follows Benhamou (2004) <[doi:10.1016/j.jtbi.2004.03.016](#)>. Renders publication-quality 'ggplot2' radial plots with overlaid rose diagrams, parametric density curves, non-parametric kernel density estimates, and circular boxplots (Buttarazzi, Pandolfo and Porzio (2018) <[doi:10.1111/biom.12889](#)>). Includes a browser-based 'shiny' 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)

Copyright file inst/COPYRIGHTS

Encoding UTF-8

Language en-GB

LazyData true

LazyDataCompression xz

Depends R (>= 4.1.0)

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

Suggests bpDir, dplyr (>= 1.0.0), testthat (>= 3.2.0), knitr,
rmarkdown, ggrepel, arrow, data.table (>= 1.14.0), jsonlite,
readr, readxl, yaml, withr, writexl, shiny (>= 1.7.0), bslib
(>= 0.5.0), pkgload, png, jpeg, svglite, shinytest2, chromote,
CircMLE, vdiff, sphunif

Config/testthat/edition 3

VignetteBuilder knitr

Collate 'conditions.R' 'Tracks.R' 'circular_boxplot.R'
'circular_mapping.R' 'circular_regression.R'
'concentration_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' 'circ_mixtures.R' 'model_selection.R'
'circumference_labs.R' 'periodic_angle.R' 'profile_plot.R'
'radiatR-package.R' 'simulate_tracks.R' 'spelling.R'
'stimulus_landmarks.R' 'time.R' 'track_metrics.R'
'transforms.R'

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author John D. Kirwan [aut, cre, cph], A. & M.W. Mathis Labs [cph]
(test fixture
tests/testthat/fixtures/real/dlc_CollectedData_Pranav.csv, from
'DeepLabCut'), Talmo Pereira [cph] (example file
inst/extdata/sleep_example.csv, from 'SLEAP'), The Trustees of
Princeton University [cph] (example file
inst/extdata/sleep_example.csv, from 'SLEAP')

Maintainer John D. Kirwan <john@jckirwan.org>

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

Date/Publication 2026-10-27 18:50:11 UTC

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

RemoteRef HEAD

RemoteSha e5f4d107f7f5408e61fbde24665b0073c5354bb3

Contents

add_angle_rose	5
add_circ	7
add_circ_interval	8
add_circ_mean	9
add_circular_boxplot	10
add_circular_density	12
add_circular_kde	14
add_critical_r	16
add_critical_v_line	17
add_heading_arrow	19
add_heading_density	20
add_heading_interval	22
add_heading_points	24
add_heading_vectors	25
add_landmark	26
add_multiple_circles	28
add_origin_point	29
add_quadrant_lines	29
add_radial_grid	30
add_stacked_headings	31
add_stimulus_arc	33
add_ticks	34
add_vonmises_density	35
add_wrappedcauchy_density	36
angular_velocity	38
apply_transform	39
as.data.frame.Tracks	40
as_angle	41
assign_colour_key	42
assign_cycle_colours	43
bin_angles	44
boot_kappa_ci	45
boot_kappa_contrast	46
boot_mean_ci	47
circ_boxplot_stats	48
circ_cor	49
circ_dispersion	51
circ_display	51
circ_model_select	52
circ_regression	54
circ_summarise	55
circ_summary	56
circ_summary_headings	57
circular_mapping	58
circumference_labs	59
compute_circ_interval	60

compute_circ_mean	61
compute_circular_density	62
concentration_regression	64
count_goal_entries	66
cpunctatus	67
cycle_colours	68
degree_labs	69
derive_coords	70
derive_headings	71
directedness_arrow	73
distance_scale	74
elapsed_seconds	75
fitted_directions	76
frame_rate	77
get_all_object_pos	78
get_tracked_object_pos	79
get_trial_limits	81
gg_traj	82
guess_columns	83
headings_frame	84
hf_accessors	85
ids	86
import_info	86
import_tracks	87
instantaneous_speed	88
launch_app	89
line_circle_intercept	90
list_heading_rules	90
list_loader_dialects	91
list_loader_formats	91
load_manifest	92
path_sinusosity	93
path_straightness	94
path_tortuosity	95
plot_profile	96
plot_speed_direction	97
plot_speed_histogram	98
pose_to_headings	99
rad_shepherd	100
radial_theme	101
radiate	101
read_tracks	107
read_tracks_dir	112
read_tracks_format	113
reference	114
register_heading_rule	115
register_loader_dialect	116
register_loader_format	117

restrict_to_circumference	118
scale_cardinal	119
scale_clock	120
scale_months	120
scale_seconds	121
sector_summary	122
set_reference	122
simulate_tracks	123
sinuosity	126
stack_headings	127
step_speed	129
straightness_index	129
test_concentration	130
test_distributions	131
test_gof	133
test_mean_directions	134
test_symmetry	135
test_uniformity	136
test_unimodality	138
tortuosity_ratio	139
track_duration	140
track_length	141
track_speed	141
track_turning	142
track_velocity	143
Tracks-class	144
transform_history	147
velocity_angle	148
velocity_vector	149
vonmises_fit	150
wrappedcauchy_fit	151
zone_dwell	152

Index**154**

add_angle_rose	<i>Add a rose diagram of heading angles to a radiate plot</i>
----------------	---

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(facets = ...)` 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 <code>derive_headings(..., rule = "bodypart_axis", frame_select = "all")</code> .
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 <code>facets</code> argument of the parent <code>radiate()</code> 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). <code>color</code> is the American-spelling alias.
alpha	Opacity. Default 0.5.
arc_pts	Points used to approximate each wedge arc. Default 20L.
display	A <code>'circ_display()'</code> controlling rotation, matching the parent <code>'radiate()'</code> plot. Default <code>'NULL'</code> uses the input's <code>'display'</code> attribute when present, otherwise the identity display.
axial	Logical; when <code>'TRUE'</code> , mirror each observation to <code>'angle_col + pi'</code> before estimation, yielding a period-pi (bidirectional) result. Default <code>'FALSE'</code> .

Value

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

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
radiate(cpunctatus[1:5]) + add_angle_rose(hd)
```

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	<i>Render a pre-computed circular interval arc on a radial plot</i>
-------------------	---

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()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
iv <- compute_circ_interval(hd, stat = "sd")
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_circ_interval(iv)
```

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()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
sm <- compute_circ_mean(hd)
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_circ_mean(sm)
```

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 = 1.1,
  width = 0.05,
  colour = "black",
  box_fill = "grey90",
  farout_shape = 8,
  show_median_arrow = FALSE,
  linewidth = 0.8,
  n_theta = 200L,
  display = NULL,
  facets = NULL,
  group_col = NULL,
  group_gap = 0.06,
  theme = 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- π): 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	Circumference radius for the box/whiskers. Default '1.1' (just outside the unit circle).
width	Radial thickness of the box band. Default '0.05'.
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 "grey90".
farout_shape	Point shape for far-out values. Default '8' (star).
show_median_arrow	Draw a centre-to-edge arrow at the median direction (a median pointer, not a rho resultant). Default 'FALSE'.
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()'.

facets	Optional character vector of column names in 'hd' identifying facet panels (the same columns passed to 'radiate(rows = , cols =)'). A single name gives one boxplot per level; two names give one per cell of the column combination (grid mode). The layer data is tagged with every facet column so each panel shows its own summary. A cell that is not drawable (fewer than 4 usable observations or a non-unique median) is skipped with a warning while the others still draw. Default 'NULL' draws a single boxplot from all of 'hd'.
group_col	Optional single column name in 'hd'. When set, one boxplot is drawn per group level as a concentric ring stepped outward from 'radius' by 'group_gap', with the box outline, whiskers, crossbars and far-out points coloured by the group. 'NULL' (default) draws a single boxplot. Combine with 'facets' to subdivide each panel. A group with fewer than 4 usable observations (or a non-unique median) is skipped with a warning.
group_gap	Radial step between consecutive group rings. Default '0.06'.
theme	Optional radiate theme name (e.g. "void", "minimal", "dark"). When set, the box, whiskers and crossbars take that theme's chrome colour (matching the circle and ticks), overriding colour.

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](https://doi.org/10.1111/biom.12889)

See Also

[circ_boxplot_stats()], [radiate()]

Examples

```
tr <- simulate_tracks(output = "trajset", seed = 1)
hd <- derive_headings(tr, rule = "distal")
radiate(tr) + add_circular_boxplot(hd)
```

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	<i>Overlay a non-parametric circular kernel density estimate on a radiate plot</i>
------------------	--

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 facets in the parent radiate() call.
bw	Bandwidth (concentration). NULL uses <code>bw.nrd.circular</code> 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 <code>density.circular</code> . Default "vonmises" (the kernel shape, not a model assumption).
colour, color	Outline colour. Default "tomato". <code>color</code> 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.

Examples

```

hd <- derive_headings(cpunctatus[1:10], rule = "distal")
radiate(cpunctatus[1:10]) + add_circular_kde(hd)

```

add_critical_r	<i>Add a critical resultant-length circle to a radiate plot</i>
----------------	---

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",
  facets = NULL,
  group_col = NULL,
  colour = "firebrick",
  linetype = "dashed",
  linewidth = 0.6,
  n_pts = 200L
)
```

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".
facets	Character vector of column names that define the facet panels (placement only). The critical radius is computed per facet cell and every one of these columns is attached to the layer data so the circles land in the right panel under <code>facet_wrap()</code> / <code>facet_grid()</code> . NULL pools all rows.
group_col	Optional single column. Splits each cell into subgroups (one circle each) and maps the circle colour to that column. NULL draws every circle in the fixed colour.
colour	Circle colour. Default "firebrick". Overridden by the colour scale when group_col is set.
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 occupied cell from `hd`. Pass the same column(s) to `facets` as you pass to `radiate(rows=)/radiate(cols=)` so one circle is drawn per panel at the radius appropriate to that panel's n . Pass `group_col` to split each cell further by colour.

Value

A `geom_path` layer, or `NULL` if no group has $n \geq 2$.

See Also

[add_heading_arrow](#), [add_circ](#)

Examples

```
hd <- derive_headings(cpunctatus[1:10], rule = "distal")
radiate(cpunctatus[1:10]) +
  add_heading_arrow(hd) +
  add_critical_r(hd)
```

`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 μ_0 . 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 μ_0 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 μ_0 exceeds c .

Usage

```
add_critical_v_line(
  hd,
  mu0,
  alpha = 0.05,
  angle_col = "heading",
  facets = NULL,
  group_col = NULL,
```

```

    show_region = FALSE,
    colour = "firebrick",
    linetype = "dashed",
    linewidth = 0.6,
    region_fill = "firebrick",
    region_alpha = 0.08,
    axial = FALSE,
    n_pts = 100L
  )

```

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".
facets	Character vector of column names used for faceting (panel placement). Each occupied combination gets its own boundary; the columns are attached to the returned data so facet_wrap/facet_grid routes each boundary to the correct panel. NULL (default) pools across facets.
group_col	Optional single column. Subdivides each facet cell into subgroups, one boundary per subgroup (each with its own n). All boundaries are drawn in the fixed colour — group_col does not map colour.
show_region	Logical; shade the rejection segment. Default FALSE.
colour	Line colour. Default "firebrick".
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 boundary from `hd`: one boundary per occupied combination of `facets` and `group_col` columns, or a single pooled boundary when both are NULL.

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](#)

Examples

```
hd <- derive_headings(cpunctatus[1:10], rule = "distal")
radiate(cpunctatus[1:10]) +
  add_heading_arrow(hd) +
  add_critical_v_line(hd, mu0 = 0)
```

add_heading_arrow	<i>Compute a circular mean arrow and add it to a radial plot in one step</i>
-------------------	--

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

<code>headings_df</code>	Data frame with a column of heading angles in radians. <code>[derive_headings()]</code> sets <code>'attr(headings_df, "angle_convention")'</code> and <code>'attr(headings_df, "coords")'</code> automatically.
<code>heading_col</code>	Name of the column containing heading angles. Default <code>"heading"</code> .
<code>colour_col, color_col</code>	Optional. Name of a column in <code>'summary_df'</code> to map to the colour aesthetic. Ignored when <code>'colour'</code> is also supplied. <code>'color_col'</code> is the American-spelling alias.
<code>display</code>	A <code>'[circ_display]'</code> object. When <code>'NULL'</code> (default), read from <code>'attr(headings_df, "display")'</code> , falling back to <code>'circ_display()'</code> .

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()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_heading_arrow(hd)
```

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",
  facets = NULL,
  group_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 = NULL
)
```

Arguments

headings_df	Data frame containing heading angles.
heading_col	Name of the heading column (radians). Default "heading".
facets	Character vector of column names used for faceting (placement). Passed to [compute_circ_interval()]; each column is attached to the layer data so 'facet_grid()'/ 'facet_wrap()' can place the arc correctly. Default 'NULL'.
group_col	Single column name used for colour grouping. Passed to [compute_circ_interval()] and then to [add_circ_interval()] as 'colour_col'. Default 'NULL'.
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()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_heading_interval(hd, stat = "sd")
```

add_heading_points *Add heading endpoint markers on the unit circle*

Description

Draws a hollow circle at $(\cos(\text{heading}), \sin(\text{heading}))$ 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

<code>headings_df</code>	Data frame with a 'heading' column (angles in radians).
<code>colour_col, color_col</code>	Name of a column in 'headings_df' to map to the colour aesthetic. When 'NULL' (default), the value of <code>attr(headings_df, "colour_col")</code> 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.
<code>colour, color</code>	Fixed colour string. Overrides 'colour_col' when supplied; when 'NULL' and no 'colour_col' resolves, defaults to "black". 'color' is the American-spelling alias.
<code>size</code>	Point size passed to 'geom_point'.
<code>alpha</code>	Point alpha transparency.
<code>axial</code>	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	<i>Add heading vector segments from inner crossing to unit circle</i>
---------------------	---

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_landmark

Add a landmark marker to a circular plot

Description

Draws a styled point just outside the rim at 'bearing', with a sun-like default (a gold asterisk), plus an optional text label. Change 'shape', 'colour'/'fill' and 'label' to mark any landmark (nest, feeder, light source). The bearing is given in **display units** (the value read off the axis), so the marker lands at the labelled position under any [circ_display()] convention.

Usage

```
add_landmark(
  bearing = 0,
  r = 1.12,
  shape = 8,
  size = 4,
  colour = "goldenrod",
  fill = "gold",
  label = NULL,
```

```

    display = circ_display(),
    ...,
    color = NULL
  )

```

Arguments

bearing	Direction of the landmark, in display units (degrees by default). Default '0'.
r	Radius at which to draw the point. Default '1.12' (just outside the unit circle).
shape	Point shape. Default '8' (asterisk, sun-ray-like).
size	Point size. Default '4'.
colour, color	Point colour. Default "goldenrod". 'color' is the American-spelling alias.
fill	Fill colour, used only by fillable shapes (21–25). Default "gold".
label	Optional text label, drawn just beyond the point. 'NULL' (default) draws no label.
display	A [circ_display()] giving the plot's angle convention. Default 'circ_display()'. Pass the same value used in [radiate()].
...	Further arguments passed to [ggplot2::geom_point()].

Value

A list of ggplot layers: a 'geom_point()' layer, plus a 'geom_text()' layer when 'label' is supplied.

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_stimulus_arc()], [add_origin_point()]

Examples

```

library(ggplot2)
ggplot() + coord_fixed() + add_circ() +
  add_landmark(bearing = 45, label = "sun")

```

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

<code>radii</code>	A vector of radii for the concentric circles (default is <code>c(0.25, 0.5, 0.75)</code>)
<code>circle_colour</code> , <code>circle_color</code>	The colour of the circles (default is "grey20"). ‘circle_color’ is the American-spelling alias.
<code>circle_alpha</code>	The transparency of the circles (default is 1)
<code>circle_size</code>	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	<i>Mark the centre of a radial plot</i>
------------------	---

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 <code>"grey50"</code> . <code>'color'</code> is the American-spelling alias.
size	Point size. Default <code>'1.5'</code> .
shape	Point shape. Default <code>'16'</code> (filled circle).
...	Further arguments passed to <code>[ggplot2::geom_point()]</code> .

Value

A `'geom_point()'` layer.

Examples

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

add_quadrant_lines	<i>Add quadrant lines to a radial plot</i>
--------------------	--

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 <code>"grey60"</code> . <code>'color'</code> is the American-spelling alias.
linewidth	Line width. Default <code>'0.5'</code> .
linetype	Line type. Default <code>"dashed"</code> .

Value

A `'geom_segment()'` layer.

Examples

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

add_radial_grid	<i>Radial grid layers (the radial analogue of a Cartesian grid)</i>
-----------------	---

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 ($\pi / \text{spokes_major}$), 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 = step/2,
  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

<code>data</code>	A data frame with an angle column in radians, typically a headings_frame .
<code>col</code>	Name of the angle column. Defaults to the <code>heading_col</code> attribute when data is a <code>headings_frame</code> .
<code>step, start_sep, tol, direction, base_r</code>	Passed to stack_headings when <code>stack_r</code> is absent. See that function for details. <code>step</code> sets the gap between dots and <code>start_sep</code> (default <code>step / 2</code>) offsets the first dot inward from the reference circle so its outer edge approximately abuts the circumference.
<code>shade</code>	If TRUE, map <code>stack_n</code> to the alpha aesthetic (scaled 0.2–1 across the observed range). Overrides the fixed alpha argument.
<code>shape</code>	Passed to stack_headings to request per-observation shape encoding. Shape is also applied when <code>shape_code</code> is already a column in data. Mapped to ggplot2 shape integers: 1 = hollow, 16 = filled, 21 = filled with ring.
<code>group</code>	Optional column name; stack within each group independently (e.g. one stacking per facet). Default NULL.
<code>colour, color</code>	Fixed point colour (ignored when <code>colour_col</code> is set). <code>color</code> is the American-spelling alias.
<code>colour_col, color_col</code>	Optional column name to map to the colour aesthetic. <code>color_col</code> is the American-spelling alias.
<code>size</code>	Point size passed to <code>geom_point()</code> .
<code>alpha</code>	Fixed alpha. Ignored when <code>shade = TRUE</code> .

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 <code>ggplot2::geom_point()</code> .

Value

A `geom_point()` layer.

See Also

[headings_frame](#), [stack_headings](#), [add_heading_points](#)

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_stacked_headings(hd)
```

add_stimulus_arc	<i>Add a stimulus arc to a circular plot</i>
------------------	--

Description

Draws a filled ribbon on the rim of the unit circle, spanning 'width' in angular extent centred on 'bearing', depicting a dark bar or spot stimulus in an orientation arena. A wide 'width' reads as a bar; a narrow 'width' as a spot. The bearing is given in **display units** (the value read off the axis), so the arc lands at the labelled position under any `[circ_display()]` convention.

Usage

```
add_stimulus_arc(
  bearing = 0,
  width = 20,
  thickness = 0.05,
  r = 1,
  fill = "black",
  colour = NA,
  alpha = 1,
  display = circ_display(),
  color = NULL
)
```

Arguments

bearing	Centre of the stimulus, in display units (degrees by default). Default '0'.
width	Angular extent of the stimulus, in display units. Default '20'.
thickness	Radial extent of the ribbon, in data units. Default '0.05' (straddles the circumference).
r	Centre radius of the ribbon. Default '1' (on the unit circle).
fill	Fill colour. Default '"black"'.
colour, color	Border colour. Default 'NA' (no border). 'color' is the American-spelling alias.
alpha	Fill alpha transparency. Default '1'.
display	A [circ_display()] giving the plot's angle convention. Default 'circ_display()'. Pass the same value used in [radiate()].

Value

A list containing one 'ggplot2::annotate("polygon")' layer.

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_landmark()], [add_circ()]

Examples

```
library(ggplot2)
ggplot() + coord_fixed() + add_circ() +
  add_stimulus_arc(bearing = 90, width = 30)
```

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

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

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

Value

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

Examples

```

hd <- derive_headings(cpunctatus[1:10], rule = "distal")
fit <- vonmises_fit(hd)
radiate(cpunctatus[1:10]) + add_vonmises_density(fit)

```

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

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

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](#)

Examples

```
hd <- derive_headings(cpunctatus[1:10], rule = "distal")
fit <- wrappedcauchy_fit(hd)
radiate(cpunctatus[1:10]) + add_wrappedcauchy_density(fit)
```

angular_velocity	<i>Per-observation angular (turning-rate) velocity for a Tracks</i>
------------------	---

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()]

Examples

```
summary(angular_velocity(cpunctatus, units = "degrees"))
```

apply_transform	<i>Apply a bespoke transformation to a Tracks</i>
-----------------	---

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.

Examples

```
head(as.data.frame(cpunctatus))
```

as_angle

Map periodic time, date, or numeric data onto circular angles

Description

Converts a vector of timestamps, dates, or a generic periodic numeric into unit-circle angles in radians ($[0, 2\pi)$), so periodic data — events by time of day (chronobiology) or time of year (phenology) — can be analysed and drawn with the circular tools ('circ_summary()', 'vonmises_fit()', 'radiate()', the distribution overlays). It is the data-side counterpart of the [circumference_labs()] / [scale_clock()] / [scale_months()] labels: the cycle start is placed at the top and time increases clockwise, so a '06:00' timestamp lands exactly on the "6" hours label and the 1st of each month on its month label.

Usage

```
as_angle(x, period)
```

Arguments

x	A 'POSIXct', 'Date', or 'numeric' vector.
period	The cycle. "day" (time of day; 'x' must be 'POSIXct'), "year" (time of year; 'Date' or 'POSIXct', day-accurate and leap-safe), or a single positive number giving the cycle length in the units of a numeric 'x'.

Value

A 'numeric' vector of angles in radians ($[0, 2\pi)$), the same length as 'x'; 'NA' entries propagate. Wrap it with [headings_frame()] (or use it as a 'heading' column) to feed the circular tools.

See Also

[circumference_labs()], [scale_clock()], [scale_months()], [headings_frame()], [circ_summary()], [radiate()]

Examples

```
# events through a day -> a clock-face circular distribution
set.seed(1)
times <- as.POSIXct("2020-06-01", tz = "UTC") +
  rnorm(200, mean = 9 * 3600, sd = 2 * 3600) # clustered around 09:00
ang <- as_angle(times, "day")
circ_summarise(data.frame(heading = ang), "heading", units = "radians")

# a generic numeric cycle of length 10
as_angle(c(0, 2.5, 5, 7.5), period = 10)
```

assign_colour_key	<i>Assign a shared colour-key column to a Tracks or data frame</i>
-------------------	--

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 <i>centre</i> . 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 <code>circular::plot.circular</code> (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)
```

boot_kappa_ci	<i>Bootstrap confidence intervals for circular concentration</i>
---------------	--

Description

Computes nonparametric bootstrap confidence intervals for the concentration of each group, for both the von Mises maximum-likelihood concentration κ and the mean resultant length R . Angles are resampled with replacement and the interval is the percentile envelope of the bootstrap estimates. Unlike the normal-approximation `se_kappa` from [vonmises_fit](#), it makes no distributional assumption and stays usable at low concentration or small sample size, where that approximation is unreliable and where $\hat{\kappa}/\hat{R}$ carry a known upward bias.

Usage

```
boot_kappa_ci(
  hd,
  group_col = NULL,
  angle_col = "heading",
  conf = 0.95,
  R = 999L,
  axial = FALSE
)
```

Arguments

hd	Data frame containing headings in radians.
group_col	Column(s) to group by. NULL (default) treats the whole data frame as one group.
angle_col	Name of the heading column. Default "heading".
conf	Confidence level. Default 0.95.
R	Number of bootstrap resamples. Default 999. Set the RNG seed with set.seed for reproducible intervals.
axial	Logical; when TRUE, treat the angles as axial (bidirectional, mod-pi): concentration is estimated in the doubled-angle frame (about the axis) and, like vonmises_fit , is not rescaled. Default FALSE (directional).

Details

The `kappa_bias/R_bias` columns report the bootstrap estimate of that bias (bootstrap mean minus the sample estimate); the intervals themselves are the raw percentile envelope and are not bias-shifted.

Value

Data frame with columns `group_col` (if supplied), `kappa` (von Mises MLE concentration), `kappa_ci_lo/kappa_ci_hi` (conf-level percentile CI on κ), `kappa_bias`, `resultant_R` (mean resultant length in $[0, 1]$), `R_ci_lo/R_ci_hi`, `R_bias`, and `n`. Groups with fewer than two finite angles yield an all-NA row.

See Also

[boot_mean_ci](#), [boot_kappa_contrast](#), [vonmises_fit](#), [circ_dispersion](#), [test_concentration](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
set.seed(1)
boot_kappa_ci(hd, group_col = "arc", R = 199)
```

<code>boot_kappa_contrast</code>	<i>Bootstrap confidence interval for a between-condition concentration contrast</i>
----------------------------------	---

Description

Contrasts the concentration of two (or more) groups via a stratified nonparametric bootstrap: angles are resampled with replacement within each group independently, and the contrast is recomputed on every resample. For a pair of groups it reports the difference in von Mises concentration ($\Delta\kappa$), the concentration ratio (κ_1/κ_2), and the difference in mean resultant length (ΔR), each with a percentile confidence interval, plus a two-sided bootstrap p-value for $\Delta\kappa = 0$. It is the assumption-light, estimation-focused companion to [test_concentration](#) (parametric `equal.kappa` / non-parametric Wallraff).

Usage

```
boot_kappa_contrast(
  hd,
  group_col,
  angle_col = "heading",
  conf = 0.95,
  R = 999L,
  axial = FALSE
)
```

Arguments

<code>hd</code>	Data frame containing headings in radians.
<code>group_col</code>	Grouping column. Required; must have at least two distinct values. With exactly two groups a single contrast row is returned; with more than two, every unordered pair is returned.
<code>angle_col</code>	Name of the heading column. Default "heading".

conf	Confidence level. Default 0.95.
R	Number of bootstrap resamples. Default 999. Set the RNG seed with set.seed for reproducible intervals.
axial	Logical; when TRUE, treat the angles as axial and estimate concentration in the doubled-angle frame (see boot_kappa_ci). Default FALSE.

Details

The concentration ratio is skewed and can be unstable or non-finite when the denominator concentration is near zero; $\Delta\kappa$ is the more robust default in the low-concentration regime.

Value

Data frame with one row per group pair: group1, group2, delta_kappa, delta_kappa_ci_lo/delta_kappa_ci_hi, kappa_ratio, kappa_ratio_ci_lo/kappa_ratio_ci_hi, delta_R, delta_R_ci_lo/delta_R_ci_hi, p_value (two-sided bootstrap p for $\Delta\kappa = 0$), n1, n2.

See Also

[boot_kappa_ci](#), [test_concentration](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "type")
set.seed(1)
boot_kappa_contrast(hd, group_col = "type", R = 199)
```

boot_mean_ci	<i>Bootstrap confidence interval for a mean direction</i>
--------------	---

Description

Computes a nonparametric bootstrap confidence interval for the mean direction of each group, resampling angles with replacement and taking the percentiles of the bootstrap mean directions (centred on the sample mean). Unlike the normal-approximation interval from [vonmises_fit](#), it makes no distributional assumption and stays usable at low concentration or small sample size, where that approximation is unreliable.

Usage

```
boot_mean_ci(
  hd,
  group_col = NULL,
  angle_col = "heading",
  conf = 0.95,
  R = 999L,
  axial = FALSE
)
```

Arguments

hd	Data frame containing headings in radians.
group_col	Column(s) to group by. NULL (default) treats the whole data frame as one group.
angle_col	Name of the heading column. Default "heading".
conf	Confidence level. Default 0.95.
R	Number of bootstrap resamples. Default 999. Set the RNG seed with set.seed for reproducible intervals.
axial	Logical; when TRUE, treat the angles as axial (bidirectional, mod- π): μ/μ_{deg} are the mean axis in $[0, \pi)$ and the interval is scaled accordingly. Default FALSE (directional).

Value

Data frame with columns group_col (if supplied), mu (mean direction, radians), mu_deg (degrees), ci_lo and ci_hi (conf-level bootstrap interval on mu, radians; bounds may fall outside $[0, 2\pi)$ to describe an arc), resultant_R, and n. Groups with fewer than two finite angles yield an all-NA interval.

See Also

[vonmises_fit](#), [circ_dispersion](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
set.seed(1)
boot_mean_ci(hd, group_col = "arc", R = 199)
```

`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 antimedial, 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- π): 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](https://doi.org/10.1111/biom.12889)

See Also

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

Examples

```
tr <- simulate_tracks(output = "trajset", seed = 1)
hd <- derive_headings(tr, rule = "distal")
circ_boxplot_stats(hd)
```

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.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal")
# does the heading depend on the frame at which it was taken?
circ_cor(hd, x_col = "time")
```

circ_dispersion	<i>Per-group circular dispersion statistics for a dense heading series</i>
-----------------	--

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(..., rule = "bodypart_axis", 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- π) 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 $\sqrt{-2 \log R}$.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
circ_dispersion(hd, group_col = "arc")
```

circ_display	<i>Circular display convention specification</i>
--------------	--

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.

Examples

```
# Default: North at top, clockwise, degrees
circ_display()

# Stimulus at East should appear at top
circ_display(zero = 0)

# Counterclockwise, radians
circ_display(clockwise = FALSE, units = "radians")
```

circ_model_select	Select among candidate circular models by AICc
-------------------	--

Description

Fits five candidate models to a heading sample and ranks them by the small-sample-corrected Akaike information criterion (AICc):

uniform no preferred direction ($k = 0$).

unimodal one von Mises mode ($k = 2$).

axial symmetric bimodal von Mises, two equal antipodal modes ($k = 2$).

unimodal_uniform a directed von Mises mode over a uniform background, $p vM(\mu, \kappa) + (1-p) U$ ($k = 3$).

bimodal an asymmetric two-component von Mises mixture with free means, concentrations, and weight ($k = 5$).

The mixture models (unimodal_uniform, bimodal) are fitted by numerical maximum likelihood; the other three are closed-form. This model set mirrors the Schnute-Groot family implemented by the **CircMLE** package (Fitak & Johnsen 2017): uniform = M1, unimodal = single von Mises, axial = symmetric bimodal, unimodal_uniform = a directed component over a uniform background, and bimodal = a free two-component mixture.

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 for uniform, unimodal, and axial 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 (five rows, or five 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 $n - k - 1 \leq 0$; remaining weights sum to 1.

References

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

Schnute, J.T. & Groot, K. (1992). Statistical analysis of animal orientation data. *Animal Behaviour*, 43(1), 15-33.

Fitak, R.R. & Johnsen, S. (2017). Bringing the analysis of animal orientation data full circle: model-based approaches with maximum likelihood. *Journal of Experimental Biology*, 220(21), 3878-3882.

See Also

[vonmises_fit](#), [test_uniformity](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc",
  coords = "relative")
# Control (no stimulus) against the widest arc
circ_model_select(hd[hd$arc %in% c("0", "50"), ], group_col = "arc")
```

circ_regression	<i>Circular-linear regression of a heading on linear covariates</i>
-----------------	---

Description

Fits the Fisher-Lee circular-linear regression $\theta_i \sim \text{vM}(\mu_0 + 2 \arctan(x'_i \beta), \kappa)$ 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

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

Examples

```
# Headings whose mean turns with x: mu0 = 1, slope 1.5
set.seed(1)
x <- runif(60, -1, 1)
noise <- as.numeric(circular::rvonmises(60, circular::circular(0), kappa = 8))
hd <- data.frame(x = x, heading = (1 + 2 * atan(1.5 * x) + noise) %% (2 * pi))
fit <- circ_regression(hd, heading ~ x)
fit
summary(fit)
predict(fit, newdata = data.frame(x = c(-1, 0, 1)))
```

circ_summarise	<i>Tidy circular summary of a grouped data frame</i>
----------------	--

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 (unit-circle convention); `mean_dir_deg` is that direction in degrees in the display convention (by default 0 = North, clockwise), so it is not `mean_dir * 180 / pi`.

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

<code>data</code>	A data frame or grouped tibble.
<code>col</code>	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). "kappa" is the von Mises maximum-likelihood estimate from the mean resultant length, with no small-sample bias correction (see circ_summary()).
display	A circ_display object giving the convention in which mean_dir_deg is reported (the direction of zero and whether angles increase clockwise). Defaults to circ_display(): 0 = North, clockwise. Anything else, NULL included, is an error (class radiatR_invalid_argument).

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 summary statistics for trajectories

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)

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

Arguments

<code>x</code>	A ['Tracks'] object.
<code>w</code>	Character. Name of a weight column in 'x@data', used in place of the weight column of 'x'. Its values must be finite and ' ≥ 0 '; a zero weight excludes an observation. When 'NULL' (default), the weight column of 'x' is used, and all steps are weighted equally when 'x' has none. A group whose weights are all zero is an error (class 'radiatR_nonestimable').
<code>by</code>	Character. "id" (default) returns one row per trial; "global" pools all observations into a single summary row.
<code>axial</code>	Logical. Treat the angles as axial (bidirectional, mod- π) data: statistics are computed via the angle-doubling method and the mean is reported as an axis in '[0, π)' radians / '[0, 180)' degrees. Default 'FALSE' (ordinary directional data).

Value

A 'data.frame' with columns 'id', 'n', 't_start', 't_end', 'mean_dir' (radians, unit-circle convention, 0 to 2π ; weighted when weights are used), 'resultant_R' (0–1), and 'kappa' (von Mises concentration: the maximum-likelihood estimate from the mean resultant length, weighted when weights are used; 'NA' for fewer than three valid angles of positive weight, 'Inf' when all angles coincide). 'n' counts the valid angles whatever their weight. No small-sample bias correction is applied, so kappa is biased upward for small n: `circular::mle.vonmises(bias = TRUE)` gives a corrected estimate and `boot_kappa_ci()` reports the bias.

Examples

```
data(cpunctatus)
head(circ_summary(cpunctatus, by = "id"))
circ_summary(cpunctatus, by = "global")
```

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, group_by = "id", ...)
```

Arguments

<code>x</code>	A ['Tracks'] object.
<code>rule</code>	Required. Heading derivation rule passed to [derive_headings()]. One of "crossing", "distal", "straight", "origin_mean", "net", or "velocity_mean".
<code>group_by</code>	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. Other columns must first be carried onto the heading table with 'carry' (e.g. 'carry = "arc"').
<code>...</code>	Additional arguments forwarded to [derive_headings()], such as 'circ0', 'circ1', 'carry', 'return_coords', or 'coords'.

Value

A 'data.frame' with grouping columns followed by 'mean_dir' (radians, unit-circle convention, 0 to 2π), 'resultant_R' (0–1), 'kappa' (von Mises concentration, with no small-sample bias correction; see [circ_summary\(\)](#)), and 'n' (number of valid headings in the group).

Examples

```
data(cpunctatus)
# per-trial headings (default)
head(circ_summary_headings(cpunctatus, rule = "crossing",
                           circ0 = 0.2, circ1 = 0.4))

# per-condition summary: carry the "arc" column through to group by it
circ_summary_headings(cpunctatus, rule = "crossing",
                      circ0 = 0.2, circ1 = 0.4,
                      carry = "arc", group_by = "arc")
```

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.

circumference_labs	<i>Label the circumference of a radial plot in domain units</i>
--------------------	---

Description

Adds circumference text labels (cardinal directions, hours, months, seconds, or a custom scale) to a radial ggplot. The display-aware sibling of [degree_labs()]: labels are positioned through the supplied 'display' so they track the plotted data when the convention changes.

Usage

```
circumference_labs(
  scale,
  display = circ_display(),
  colour = "black",
  size = 3.88,
  family = "",
  radius = 0.85,
  color = NULL
)
```

Arguments

scale	A circumference-scale list, from [scale_cardinal()], [scale_clock()], [scale_months()], [scale_seconds()], or hand-written as 'list(n =, at =, labels =)' ('at' in raw unit-circle radians, 0 = East, counterclockwise; 'labels' the same length as 'at').
display	A [circ_display()] convention. Default 'circ_display()' (North/top, clockwise).
colour, color	Label colour. Default '"black"'. 'color' is the American-spelling alias.
size	Label text size, in mm. Default '3.88' (ggplot2's default).
family	Label font family. Default '""' (device default).
radius	Radial distance of the labels from the centre, as a multiple of the unit circle. Default '0.85'.

Value

A list of ggplot2 annotation layers, add to a plot with '+'.

See Also

[degree_labs()] for degree/radian labels; [scale_cardinal()], [scale_clock()], [scale_months()], [scale_seconds()].

Examples

```
library(ggplot2)
ggplot() +
  coord_fixed() +
  circumference_labs(scale_months())
```

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",
  facets = NULL,
  group_col = NULL,
  stat = c("bootstrap_ci", "sd"),
  boot_reps = 1000L,
  boot_alpha = 0.05,
  axial = FALSE
)
```

Arguments

headings_df	Data frame containing heading angles.
heading_col	Name of the heading column (radians). Default "heading".
facets	Character vector of column names used for faceting (placement). One row is returned per occupied combination of 'facets' and 'group_col'. Each column is preserved in the output so 'facet_grid()/'facet_wrap()' can place the layer correctly. Default 'NULL'.
group_col	Single column name used for colour grouping. When set, one row is returned per group and the column is preserved in the output for colour mapping in [add_circ_interval()]. Default 'NULL'.
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()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
# Fast: +/-1 circular SD
compute_circ_interval(hd, stat = "sd")
```

compute_circ_mean	<i>Compute circular mean direction and resultant length from a headings data frame</i>
-------------------	--

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",
  facets = NULL,
  group_col = NULL,
  axial = FALSE
)
```

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".
facets	Character vector of column names used as faceting variables. One row is returned per unique combination of these columns. These columns are attached to the output so that [add_circ_mean()] can route each arrow to the correct facet panel.

group_col	Optional. Name of a single column to group by for colour mapping. One row is returned per unique combination of 'c(facets, group_col)'. Map colour in [add_circ_mean()] via its 'colour_col' argument.
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), plus any 'facets'/'group_col' columns. Both statistics are 'NA' when a cell contains fewer than 2 finite angles.

See Also

[add_circ_mean()], [add_heading_arrow()]

Examples

```
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
compute_circ_mean(hd)
```

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 'facets'.

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):
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()

# 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)
```

concentration_regression

Von Mises regression of concentration on covariates

Description

Fits a von Mises model in which the concentration κ varies with one or more covariates while the mean direction μ is a single constant: $\theta_i \sim \text{vM}(\mu, \kappa_i)$ with $\kappa_i = h(x_i' \gamma)$ for an inverse link h . It is the concentration-regression complement to [circ_regression](#) (which models the mean with a constant concentration), and the estimator that pairs with [simulate_tracks](#)'s `concentration_slope`. The model is fitted by maximum likelihood with [optim](#).

Usage

```
concentration_regression(
  data,
  formula,
  link = c("log", "identity"),
  init = NULL
)

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

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

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

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

Arguments

data	A data frame containing the response and predictor columns.
formula	A formula heading $\sim x_1 + x_2$; the LHS is the angle column (radians), the RHS the covariate(s) modelling κ . The intercept is retained (it is the baseline concentration on the link scale).
link	Link relating κ to the linear predictor: "log" (default; $\kappa = e^{x'\gamma}$, always positive) or "identity" ($\kappa = x'\gamma$, floored at a small positive value).
init	Optional numeric starting values for the concentration coefficients (length = number of design-matrix columns, intercept first).
object	A concentration_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 concentration_regression object.

Value

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

References

Mardia, K. V. & Jupp, P. E. (2000). *Directional Statistics*. Wiley.

See Also

[circ_regression](#), [boot_kappa_ci](#), [test_concentration](#), [simulate_tracks](#)

Examples

```
# Does orientation sharpen as the stimulus arc widens?
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc",
  coords = "relative")
hd$arc_deg <- as.numeric(as.character(hd$arc))
fit <- concentration_regression(hd, heading ~ arc_deg)
fit
summary(fit)
# Fitted kappa for a narrow, medium and wide arc
predict(fit, newdata = data.frame(arc_deg = c(5, 25, 50)))
```

count_goal_entries	<i>Count entries into a goal zone for trajectories in a circular field</i>
--------------------	--

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

```
# Treat 45 degrees (NE) at the wall as the goal, as for a water maze probe
data(cpunctatus)
entries <- count_goal_entries(cpunctatus, target_angle = pi / 4,
                             crossing_radius = 0.15)
# n_entries > 1 indicates memory of the platform location
head(entries)
```

cpunctatus	<i>*Cyldroiulus punctatus* visual orientation trajectory dataset</i>
------------	--

Description

Pre-processed ‘Tracks’ of 235 millipede trajectories from a visual-acuity experiment. **Cyldroiulus 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 (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 as one archive, ‘inst/extdata/tracks.tar.gz’ (unpack it with ‘`untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"), exdir = tempdir())`’), 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](https://doi.org/10.1016/j.visres.2019.09.003)

Examples

```
cpunctatus
# One heading per trial (the bearing of its furthest point from the
# origin), keeping the target half-width
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
table(hd$arc)
```

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	<i>Label the four diagonal directions.</i>
-------------	--

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 = "",  
  position = c("outside", "inside", "split"),  
  inside_radius = 0.88,  
  n = NULL,  
  outer_radius = NULL,  
  color = NULL  
)
```

Arguments

display	A [<code>'circ_display'</code>] object. Default <code>'circ_display()'</code> . Supplies the label units when <code>'units'</code> is <code>'NULL'</code> .
colour, color	Label colour. Default <code>"black"</code> . <code>'color'</code> is the American-spelling alias.
units	<code>"degrees"</code> (e.g. <code>'45°'</code>) or <code>"radians"</code> (e.g. $\pi/4$). When <code>'NULL'</code> (default) the units are taken from <code>'display'</code> .
size	Label text size, in mm. Default <code>'3.88'</code> (ggplot2's default text size).
family	Label font family. Default <code>""</code> (the device default).
position	Where to place the labels: <code>"outside"</code> (default) keeps the four diagonals (45/135/225/315) in the plot corners, as before; <code>"inside"</code> puts all eight directions (including the cardinals 0/90/180/270) just inside the circle at <code>'inside_radius'</code> ; <code>"split"</code> puts the cardinals inside and the diagonals outside. The <code>"inside"/"split"</code> modes honour <code>'display'</code> (the labels rotate with the plot).
inside_radius	Radius (unit-circle fraction) for inside labels. Default <code>'0.88'</code> .
n	Number of equally-spaced labels at <code>'360 / n'</code> degrees from 0. <code>'NULL'</code> (default) keeps the legacy set (4 diagonals for <code>"outside"</code> , 8 for <code>"inside"/"split"</code>). <code>'0'</code> draws none. For <code>'position = "split"'</code> , <code>'n'</code> must be divisible by 4 (so the quadrant directions are present); otherwise it falls back to <code>"inside"</code> with a message.
outer_radius	Optional corner magnitude for the diagonal <code>"outside"</code> labels. <code>'NULL'</code> (default) keeps the historical <code>'0.9*sqrt(2)'</code> distance; a larger value pushes them outward (e.g. to clear stacked circular-boxplot rings).

Value

A list of ggplot2 annotation layers.

Examples

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

derive_coords	<i>Derive polar and reference-relative coordinates from unit-circle position</i>
---------------	--

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	<i>Derive heading angle(s) from trajectories using specified rule</i>
-----------------	---

Description

Derive heading angle(s) from trajectories using specified rule

Usage

```
derive_headings(x, rule, ..., coords = c("absolute", "relative"))

## S4 method for signature 'Tracks'
derive_headings(
  x,
  rule,
  ...,
  first_only = FALSE,
  carry = NULL,
  on_missing = c("warn", "error", "quiet"),
  coords = c("absolute", "relative")
)
```

Arguments

<code>x</code>	Tracks
<code>rule</code>	Required. A single rule name, or a function. Built-in rules: "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"; a name registered with [register_heading_rule()] (see [list_heading_rules()]); or a function called once per trajectory as 'f(df, cols, ...)' that returns columns 'id', 'time' and 'heading'.
<code>...</code>	rule-specific options (see Details), including 'return_coords'. A built-in rule stops with an error naming any option it does not read; registered and function rules receive '...' unchecked.
<code>coords</code>	Character. Which Cartesian columns to use: "absolute" (default, uses whatever 'x'/'y' the 'Tracks' currently holds in 'Tracks@cols' – calibrated only if the caller supplied or normalized them that way) or "relative" (uses 'rel_x'/'rel_y', the landmark-rotated frame; errors if not registered).
<code>first_only</code>	logical; if TRUE, return only the first matching heading per trajectory
<code>carry</code>	optional character vector of columns from the source data to append via nearest time
<code>on_missing</code>	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' and 'n_missing' (headings, i.e. rows), 'n_trials' (distinct trials) and 'missing_ids' (the trials with at least one missing heading) 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. The 'distal' rule accepts 'max_radius' (default 'Inf'): the furthest-point search is restricted to positions with radius ' $\leq$ max_radius', so 'max_radius = 1' ignores beyond-circumference ($\rho > 1$) tracking outliers. The 'straight' rule takes 'tol' (default 'pi/18', the largest turn between successive steps) and 'min_len' (default '5', the fewest steps in a run). A step of zero length (a pause) has no direction: turns are measured between successive moving steps, so a pause neither breaks a run nor counts toward 'min_len'. The heading is the direction of the run's first moving step. 'pca_axis' takes 'source = "position"' (default; the principal axis of the positions about their centroid) or "velocity" (the principal axis of the steps about zero, which weights a step by its squared length; compare 'velocity_axis', which weights by length).

The rules that average directions take a weighting through ...:

- `velocity_mean` and `vm_fit` average the directions of the steps of a track. `weight_by = "uniform"` (default) counts every step once; `"step_length"` weights a step by its length,

which gives the direction from the first position to the last, as `net` does. The two rules return the same heading: the maximum-likelihood mean direction of a von Mises distribution is the mean direction.

- `velocity_axis` takes the same `weight_by`, with `"step_length"` as its default.
- `origin_mean` averages the bearings of the positions from the origin, weighted by radius to the power `r_power` (default 0, every position counted once; 1 gives the bearing of the centroid).

A step of zero length and a position at the origin have no direction and are left out of these averages, whatever the weighting. A track with no direction left, or whose directions cancel, has heading NA.

The other rules likewise give NA where a displacement has zero length: a track that never moves (`net`, `straight`, `goal_bias`, `maxspeed_window`, `window_net`), one that ends where it started (`net`), a window whose ends coincide (`window_net`), positions with no spread, or steps none of which moves (`pca_axis`), a `pca_axis` track whose net displacement has no component along its axis (the axis then has no sign), and a track that stays at the origin (`distal`). `window_net` and `maxspeed_window` shrink their window to fit a track shorter than `window_n`.

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`.

Examples

```
h <- derive_headings(cpunctatus, rule = "distal")
head(h)
# ignore tracking outliers beyond the circumference and keep the point used
head(derive_headings(cpunctatus, rule = "distal", max_radius = 1,
                     return_coords = TRUE))
# per-trial reference frame, carrying a condition column for grouping
head(derive_headings(cpunctatus, rule = "net", coords = "relative",
                     carry = "arc"))
```

<code>directedness_arrow</code>	<i>Make mean resultant length arrow</i>
---------------------------------	---

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

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

Value

A ‘geom_segment()’ layer.

Examples

```

# Arrow showing mean direction and resultant length
hd <- derive_headings(cpunctatus[1:5], rule = "distal")
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  directedness_arrow(hd, angle_col = "heading")

```

distance_scale	<i>Distance calibration for a Tracks object</i>
----------------	---

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-circle) 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()]

Examples

```

distance_scale(cpunctatus) # NULL: not calibrated
x <- set_distance_scale(cpunctatus, 150, unit = "mm")
distance_unit(x)
# the same scale from a known separation: 0.5 coordinate units is 75 mm
x <- calibrate_distance(cpunctatus, coord_distance = 0.5,
                        real_distance = 75, unit = "mm")

distance_scale(x)
head(track_length(x))

```

elapsed_seconds	<i>Elapsed time per observation of a Tracks object</i>
-----------------	--

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, and so is numeric time loaded as seconds ('read_tracks(time_type = "seconds")', or a duration column): a frame rate is neither needed nor applied to it. Any other numeric time is a frame index and 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()]

Examples

```
head(elapsed_seconds(cpunctatus))
```

fitted_directions	<i>Fitted mean directions from a circular regression, for plotting</i>
-------------------	--

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 $\text{circular::A1}(\kappa)$, 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()]

Examples

```
set.seed(1)
x <- runif(60, -1, 1)
noise <- as.numeric(circular::rvonmises(60, circular::circular(0), kappa = 8))
hd <- data.frame(x = x, heading = (1 + 2 * atan(1.5 * x) + noise) %% (2 * pi))
fit <- circ_regression(hd, heading ~ x)
arrows <- fitted_directions(fit, at = c(-1, -0.5, 0, 0.5, 1))
arrows
library(ggplot2)
ggplot() +
  coord_fixed() +
  add_circ() +
  add_circ_mean(arrows, colour_col = "x")
```

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. It converts frame indices only: time that is already real (POSIXct, or numeric seconds loaded with ‘read_tracks(time_type = "seconds")’) is never divided by it.

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()]

Examples

```
frame_rate(cpunctatus)
x <- set_frame_rate(cpunctatus, 25)
frame_rate(x)
```

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. Supply 'manifest' to join per-file metadata (matched on 'file') onto every trial.

Usage

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

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()].
track_dir	Directory containing the landmark and track text files.

manifest	Optional data frame (e.g. from [import_info()]) with a 'file' column matching 'file_tbl\$basename'; its other columns are joined onto 'file_tbl' and carried onto every trial.
manifest_cols	Optional named character vector (or list) mapping new column names to columns of 'manifest'. When 'NULL', all columns aside from 'file' are carried over with the same names.

Value

A 'Tracks' combining the normalised observations for all valid trials in the manifest. The aggregated trial limits are available via 'meta\$trial_limits', which holds every trial of every file, with 'valid_track' 'FALSE' for those dropped. Each landmark and track file is tab-separated with frame, x and y in the first three fields of every line (further fields are not read); a cell that is not a number stops with a 'radiatR_malformed_values' error, and a line with fewer than three fields or an odd number of landmark rows with a 'radiatR_malformed_file' error, each naming the file. When no file holds a valid trial the call stops with a 'radiatR_no_valid_trials' error.

Examples

```
untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
file_tbl <- import_tracks(track_dir)[1:3, ]
manifest <- import_info(system.file("extdata", "millipede_trials.csv",
                                   package = "radiatR"))
manifest <- manifest[manifest$file %in% file_tbl$basename, ]
ts <- get_all_object_pos(file_tbl = file_tbl, track_dir = track_dir,
                        manifest = manifest)

ts
ts@meta$trial_limits[, c("video", "arc", "type", "ref_theta")]
```

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"),
    .report_oob = TRUE
  )

```

Arguments

<code>trial_limits</code>	Data frame produced by <code>[get_trial_limits()]</code> .
<code>track</code>	Data frame or ‘Tracks’ of Cartesian coordinates for the entire video.
<code>circ0</code>	Inner radius threshold (default ‘0.1’).
<code>circ1</code>	Outer radius threshold (default ‘0.2’).
<code>radius_criterion</code>	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. With “first_past”, a radius no point reaches gives that landmark ‘NA’, with a warning.
<code>.report_oob</code>	Internal. When ‘TRUE’ (default), emit a single message summarising any track points that fall outside the unit circle. Callers that loop over many files (e.g. <code>[get_all_object_pos()]</code>) set this to ‘FALSE’ and read the tally from ‘meta\$oob_points’/‘meta\$oob_trials’ to report once.

Value

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

Examples

```

untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
file_tbl <- import_tracks(track_dir)[1, ]
read_xy <- function(f) {
  d <- utils::read.delim(file.path(track_dir, f), header = FALSE)[1:3]
  stats::setNames(d, c("frame", "x", "y"))
}
landmarks <- read_xy(file_tbl$landmark) # origin, then reference, per trial
track <- read_xy(file_tbl$track)
limits <- get_trial_limits(landmarks, track, file_tbl, vid_num = 1)
get_tracked_object_pos(limits, track)

```

get_trial_limits	<i>Summarise per-trial metadata for a single video.</i>
------------------	---

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()]. Any columns beyond 'basename', 'landmark' and 'track' are carried onto the trial table.
vid_num	Index of the current video within 'file_tbl'.

Value

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

Examples

```
untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
file_tbl <- import_tracks(track_dir)[1, ]
read_xy <- function(f) {
  d <- utils::read.delim(file.path(track_dir, f), header = FALSE)[1:3]
  stats::setNames(d, c("frame", "x", "y"))
}
landmarks <- read_xy(file_tbl$landmark) # origin, then reference, per trial
track <- read_xy(file_tbl$track)
limits <- get_trial_limits(landmarks, track, file_tbl, vid_num = 1)
limits
```

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,
  facets = 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,
  facets = 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’.
facets	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

Examples

```
# Overlay trajectories for the first two trials
gg_traj(cpunctatus[1:2])

# Facet by trial, cartesian coordinates
gg_traj(cpunctatus[1:4], facets = "trial_id", coord = "cartesian")
```

guess_columns	<i>Guess the role of each column in a track table</i>
---------------	---

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	<i>Construct a headings frame from a data frame of angles</i>
----------------	---

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

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

Examples

```
d <- data.frame(trial = 1:5, angle = c(10, 45, 80, 350, 20))
hf <- headings_frame(d, angle, units = "degrees")
hf
```

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'.

Examples

```
hf <- headings_frame(data.frame(h = c(0.1, 1.2, 3)), h, units = "radians")
hf_heading_col(hf)
hf_coords(hf)
hf_display(hf)
# a plain data frame gives the defaults
hf_heading_col(data.frame(h = 1))
```

ids	<i>Trajectory identifiers of a Tracks</i>
-----	---

Description

Trajectory identifiers of a Tracks

Usage

```
ids(x)

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

Arguments

x A [Tracks-class] object.

Value

The unique trajectory ids, in first-appearance order.

Examples

```
ids(cpunctatus)
```

import_info	<i>Import landmark coordinates from text files</i>
-------------	--

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

```
manifest_path <- system.file("extdata", "millipede_trials.csv",
                             package = "radiatR")
manifest <- import_info(manifest_path, cond_cols = c("type", "arc"))
head(manifest)
```

import_tracks	<i>Discover dtrack (or compatible) landmark/track file pairs in a directory</i>
---------------	---

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. ‘vignette("loaders")’ registers a custom loader that reads an individual trajectory file.

Value

A tibble with columns basename, landmark, and track.

Examples

```

untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
file_tbl <- import_tracks(track_dir)
head(file_tbl)

```

instantaneous_speed	<i>Per-observation instantaneous speed for a Tracks</i>
---------------------	---

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 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()]

Examples

```
summary(instantaneous_speed(cpunctatus))
```

launch_app	<i>Launch the radiatR Shiny companion app</i>
------------	---

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 is hosted on Posit Connect Cloud (git-backed) and can be deployed to any Shiny-compatible server. For a one-off manual deploy:

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

Value

Called for its side-effect; returns the result of [runApp](#) invisibly.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {  
  launch_app()  
}
```

`line_circle_intercept` *Find the intercept of a line with the unit circle*

Description

Vectorised over its arguments: each `'(x0, y0)'` -> `'(x1, y1)'` pair gives one row of the result.

Usage

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

Arguments

<code>x0, y0</code>	Starting coordinates of the vector.
<code>x1, y1</code>	Ending coordinates of the vector.

Value

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

Examples

```
# A ray from the origin through (2, 0) hits the unit circle at (1, 0)
line_circle_intercept(0, 0, 2, 0)

# Vectorised: multiple rays at once
line_circle_intercept(c(0, 0), c(0, 0), c(2, 0), c(0, 3))
```

`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](#)

Examples

```
list_heading_rules()
```

```
list_loader_dialects
```

List registered loader dialects

Description

List registered loader dialects

Usage

```
list_loader_dialects()
```

Value

A sorted character vector of registered dialect names.

Examples

```
# Built-in dialects plus any registered this session
list_loader_dialects()
```

```
list_loader_formats
```

List registered declarative formats

Description

List registered declarative formats

Usage

```
list_loader_formats()
```

Value

A sorted character vector of registered format names.

Examples

```
# Empty until a format is registered with register_loader_format()
list_loader_formats()
```

load_manifest	<i>Load trajectories listed in a file table into a Tracks</i>
---------------	---

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

Arguments

<code>file_tbl</code>	Tibble returned by <code>[import_tracks()]</code> , containing at least the columns ‘basename’ and ‘track’.
<code>track_dir</code>	Directory containing the track files on disk.
<code>manifest</code>	Optional data frame with a ‘file’ column that matches ‘file_tbl\$basename’.
<code>manifest_cols</code>	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.
<code>mapping</code>	Optional explicit column mapping passed to <code>[read_tracks()]</code> .
<code>angle_unit, time_type, tz, fps</code>	Passed to <code>[read_tracks()]</code> .
<code>normalize_xy</code>	Logical; normalise x/y to unit circle. Default FALSE. See <code>[tracks()]</code> for why TRUE is shape-preserving only, not calibration.
<code>dialect</code>	Optional dialect name; passed to <code>[read_tracks()]</code> .
<code>keep, drop</code>	Column selection vectors passed to <code>[read_tracks()]</code> .
<code>...</code>	Additional arguments forwarded to <code>[read_tracks()]</code> .

Value

A ‘Tracks’ with metadata columns replicated for each observation.

Examples

```
untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
# dtrack writes headerless, tab-separated frame, x, y (and a flag) columns
dtrack_loader <- function(x) {
  d <- utils::read.delim(x, header = FALSE, colClasses = "numeric")
  data.frame(id = tools::file_path_sans_ext(basename(x)),
             frame = d[[1]], x = d[[2]], y = d[[3]])
}
register_loader_dialect("dtrack", dtrack_loader, overwrite = TRUE)
file_tbl <- import_tracks(track_dir)[1:3, ]
manifest <- import_info(system.file("extdata", "millipede_trials.csv",
                                   package = "radiatR"))
manifest <- manifest[manifest$file %in% file_tbl$basename, ]
ts <- load_manifest(file_tbl, track_dir, manifest = manifest,
                  manifest_cols = c(arc = "arc", type = "type"),
                  dialect = "dtrack")

ts
unique(ts@data[, c("id", "arc", "type")])
```

path_sinusosity

Sinusosity index for a single trajectory

Description

The sinusosity index of Benhamou (2004), a turning-angle-based measure of path tortuosity that — unlike the displacement-based [path_straightness()] / [path_tortuosity()] — does not rely on net start-to-end displacement, so it is well-behaved for convoluted or random-search paths:

$$S = 2 \left[p \frac{1+c}{1-c} + b^2 \right]^{-1/2}$$

where p is the mean step length, c the mean cosine of the interior turning angles, and b the coefficient of variation of step length. A perfectly straight path gives ‘0’; more winding paths give larger values.

Usage

```
path_sinusosity(x, y)
```

Arguments

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

Details

Unlike the straightness index, sinuosity is **not** scale-invariant: it has units of $1/\sqrt{\text{length}}$. For the most reliable comparison across trajectories the path should be resampled to a common step length (rediscretisation); the b term mitigates variable step length but does not fully replace it.

Value

A single sinuosity value `'>= 0'`, or `'NA_real_'` when fewer than three finite points are available or all steps have zero length.

References

Benhamou, S. (2004). How to reliably estimate the tortuosity of an animal's path. *Journal of Theoretical Biology* 229(2), 209–220. doi:[10.1016/j.jtbi.2004.03.016](https://doi.org/10.1016/j.jtbi.2004.03.016)

See Also

[sinuosity()] for a whole 'Tracks'; [path_straightness()], [path_tortuosity()].

Examples

```
path_sinuosity(x = 0:5, y = rep(0, 6))           # straight -> 0
```

path_straightness	<i>Path straightness index for a single trajectory</i>
-------------------	--

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	<i>Tortuosity ratio for a single trajectory</i>
-----------------	---

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 – the non-circular companion to `[radiate()]`. ‘metric = "speed"’ plots `[instantaneous_speed()]` (one line per track); ‘metric = "turning"’ plots `[angular_velocity()]` (one line per track); ‘metric = "direction"’ plots `[velocity_angle()]` (the movement direction) as points.

Usage

```
plot_profile(
  ts,
  metric = c("speed", "turning", "direction"),
  units = c("radians", "degrees"),
  colour_by = NULL,
  facets = NULL,
  max_speed = NULL,
  smooth = 1L,
  show_raw = FALSE
)
```

Arguments

ts	A ‘Tracks’.
metric	“speed” (default), “turning”, or “direction”.
units	For ‘metric = "turning"’ (per second) or ‘metric = "direction"’, “radians” (default) or “degrees”.
colour_by, facets	Optional column names of ‘as.data.frame(ts)’ to colour by / facet into panels. Default: one neutral series per track.
max_speed	For ‘metric = "speed"’, the speed-axis cap: ‘NULL’ (default, the 99.5 plot), a positive number (hard cap), or ‘Inf’ (no clip). Off-scale points are reported in a caption. Ignored for “turning”/“direction”.
smooth	Integer window, in observations, for a centered per-track moving average of the “speed”/“turning” series. ‘1’ (default) leaves the raw series unchanged; larger values smooth out per-frame jitter. Ignored for “direction” (a circular metric).
show_raw	Logical; when ‘TRUE’ and ‘smooth > 1’, draw the raw series as a faint line beneath the smoothed line. Default ‘FALSE’.

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. The speed axis is robustly clipped by default (see 'max_speed').

Direction is circular: it is drawn as points (not a line) because a line would draw a false vertical bar across the $0/2\pi$ seam. A rotating track shows points spread across the range, and values near $0/2\pi$ ($0/360$ degrees) appear at both extremes – the honest representation.

Value

A 'ggplot2' object.

See Also

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

Examples

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

plot_speed_direction *Speed-vs-direction scatter for a Tracks*

Description

Plots each observation's instantaneous speed ([instantaneous_speed()]) against its movement direction ([velocity_angle()]) – the bivariate companion to the time profiles of [plot_profile()]. Points, since direction is circular.

Usage

```
plot_speed_direction(
  ts,
  units = c("radians", "degrees"),
  colour_by = NULL,
  max_speed = NULL
)
```

Arguments

ts	A 'Tracks'. Needs a frame rate for frame-indexed time ([set_frame_rate()]); POSIXct time is used directly.
units	Direction units: "radians" (default, $[0, 2\pi)$) or "degrees" ($[0, 360)$).
colour_by	Optional column of 'as.data.frame(ts)' to colour points by.
max_speed	Speed-axis cap: 'NULL' (default, 99.5 number (hard cap), or 'Inf' (no clip).

Details

The speed axis is robustly clipped by default ('max_speed = NULL' zooms to the 99.5 display; a caption reports how many points are off-scale. The clip is a view zoom ('coord_cartesian'), so no data are dropped. Set 'max_speed' to a number for a hard cap, or 'Inf' for the full raw range.

Value

A 'ggplot2' object.

See Also

[plot_profile()], [instantaneous_speed()], [velocity_angle()]

Examples

```
ts <- set_frame_rate(cpunctatus, 30)
plot_speed_direction(ts)
```

plot_speed_histogram *Speed distribution histogram for a Tracks*

Description

A pooled histogram of step speeds ([instantaneous_speed()]) across all tracks, annotated with the pooled median and coefficient of variation ($CV = sd / mean$, a movement-regularity descriptor). The companion distribution view to the time/scatter plots of [plot_profile()] / [plot_speed_direction()].

Usage

```
plot_speed_histogram(ts, max_speed = NULL, bins = 30)
```

Arguments

ts	A 'Tracks'. Needs a frame rate for frame-indexed time ([set_frame_rate()]); POSIXct time is used directly.
max_speed	Speed-axis cap: 'NULL' (default, 99.5 number (hard cap), or 'Inf' (no clip).
bins	Number of bars across the (clipped) speed axis. Default '30'.

Details

The speed axis is robustly clipped by default ('max_speed = NULL' zooms to the 99.5 bulk of the distribution; a caption reports how many steps are off-scale. The clip is a view zoom ('coord_cartesian'), so no data are dropped. Set 'max_speed' to a number for a hard cap, or 'Inf' for the full raw range.

Value

A 'ggplot2' object.

See Also

[instantaneous_speed()], [plot_profile()], [plot_speed_direction()]

Examples

```
ts <- set_frame_rate(cpunctatus, 30)
plot_speed_histogram(ts)
```

pose_to_headings	<i>Derive per-frame headings from pose data without a Tracks</i>
------------------	--

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

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

Value

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

Examples

```
pose <- data.frame(
  id      = "fish1", frame = 1:4,
  head_x = c(0.1, 0.2, 0.2, 0.1), head_y = c(0.0, 0.1, 0.2, 0.3),
  tail_x = c(0.0, 0.1, 0.2, 0.2), tail_y = c(0.0, 0.0, 0.1, 0.2)
)
pose_to_headings(pose, anterior = "head", posterior = "tail")
```

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

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"),
  coords = c("relative", "absolute"),
  facets = NULL,
  rows = NULL,
  cols = 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", "cardinal", "hours", "months",
    "seconds"),
  angle_label_position = c("outside", "inside", "split"),
  angle_label_outer_radius = NULL,
  n_labels = NULL,
  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,
    clip_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,
  start_sep = step/2,
  tol = NULL,
  direction = "inward",
  base_r = 1,
  shade = FALSE,
  shape = FALSE,
  facets = NULL,
  ncol = NULL,
  ticks = TRUE,
  degrees = TRUE,
  angle_labels = c("degrees", "none", "radians", "cardinal", "hours", "months",
    "seconds"),
  angle_label_position = c("outside", "inside", "split"),
  angle_label_outer_radius = NULL,
  n_labels = NULL,
  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

<code>data</code>	Data frame or ‘Tracks’.
<code>...</code>	Additional arguments forwarded to the track geom.
<code>x_col</code>	Name of the x-coordinate column. Default <code>"rel_x"</code> .
<code>y_col</code>	Name of the y-coordinate column. Default <code>"rel_y"</code> .
<code>geom</code>	Geom specification: <code>"path"</code> , <code>"point"</code> or a ggplot2 geom function.
<code>group_col</code>	Optional column for grouping aesthetics.
<code>colour_col, color_col</code>	Optional column for colour aesthetics. Mutually exclusive with ‘colour_cycle’. ‘color_col’ is the American-spelling alias.
<code>colour_cycle, color_cycle</code>	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. <code>c("red", "blue", "green")</code>). When ‘facets’ is set the cycle restarts independently within each panel. Mutually exclusive with ‘colour_col’. ‘color_cycle’ is the American-spelling alias.
<code>track_colour, track_color</code>	How trajectory paths are coloured. <code>"trajectory"</code> (default) keeps the existing per-track colouring. <code>"sequence"</code> 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 <code>"start -> finish"</code> colourbar. <code>"time"</code> colours each path by real elapsed time from its own track’s start (see <code>[elapsed_seconds()]</code>), applying a continuous viridis scale with an <code>"elapsed (s)"</code> colourbar; numeric (frame) time requires a <code>[frame_rate()]</code> (set one with <code>set_frame_rate(ts, fps)</code>). <code>"speed"</code> colours each path by its instantaneous speed (see <code>[instantaneous_speed()]</code>), applying a continuous viridis scale with a <code>"speed (units/s)"</code> colourbar; numeric (frame) time likewise requires a <code>[frame_rate()]</code> . <code>"sequence"</code> , <code>"time"</code> , and <code>"speed"</code> 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.
<code>time_units</code>	Units for ‘track_colour = "time"’: <code>"seconds"</code> (default), <code>"minutes"</code> , or <code>"milliseconds"</code> . Sets the colourbar title and scale.
<code>coords</code>	Which reference frame to plot: <code>"relative"</code> (default) uses the landmark-relative position (<code>'rel_x'/'rel_y'</code>) and relative headings; <code>"absolute"</code> uses the unit-circle position (<code>'x'/'y'</code>) in the original (un-rotated) frame and absolute headings, useful as an experimental control. Ignored when explicit <code>'x_col'/'y_col'</code> are supplied. Errors if <code>"absolute"</code> is requested but the ‘Tracks’ has no <code>'x'/'y'</code> columns registered.

facets	NULL, a column name, or a character vector of column names to facet by (via <code>[ggplot2::facet_wrap()]</code>). The named column(s) must be present in the data.
rows, cols	Optional character vectors of column names to facet into a row x column grid via <code>[ggplot2::facet_grid()]</code> (mirroring <code>ggplot</code> : 'rows' and 'cols' map to the two sides of the 'rows ~ cols' formula, and multiple variables per side nest as in 'facet_grid()'). Use these *or* 'facets' ('facet_wrap'), not both. Scales are fixed (circular panels stay round); 'ncol'/'strip_position' = "inside" apply to 'facets' only.
ncol	Number of columns passed to <code>[ggplot2::facet_wrap()]</code> when 'facets' is set.
strip_labels	Logical or 'NULL'. Whether to show a label identifying the panel variable value on each panel. Defaults to 'TRUE' when 'facets' is set, 'FALSE' otherwise. Ignored when 'facets' 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	Circumference labels. "degrees" (default; e.g. '45°'), "radians" (e.g. $\pi/4$), or "none" use diagonal angle labels via <code>[degree_labs()]</code> . The domain scales "cardinal" (N/E/S/W), "hours" (24-hour clock), "months" (Jan...Dec), and "seconds" instead label the circumference via <code>[circumference_labs()]</code> , aligning the tick count to the scale. For finer control (8-point compass, 12-hour clock, month formats) add an explicit 'circumference_labs()' layer. Label styling follows the chosen 'theme's 'axis.text'.
angle_label_position	For the numeric ("degrees"/"radians") labels, where to place them: "outside" (default) keeps the diagonals in the plot corners; "inside" puts all eight directions (including cardinals) just inside the circle; "split" puts the cardinals inside and the diagonals outside. Ignored by the domain scales ("cardinal"/"hours"/"months"/"seconds"), which place labels via <code>[circumference_labs()]</code> . See <code>[degree_labs()]</code> .
angle_label_outer_radius	Optional corner magnitude for the numeric diagonal labels ("degrees"/"radians"). 'NULL' (default) keeps the standard position; a larger value pushes the labels outward to clear wide/stacked circumference overlays such as grouped circular boxplots.
n_labels	Number of equally-spaced circumference divisions (ticks and numeric labels) at '360 / n' degrees, for the numeric "degrees"/"radians" labels. 'NULL' (default) keeps the standard 8 ticks and legacy label set; '0' draws a bare circle (no ticks or labels); e.g. '4' (cardinals), '12' (clock). Orthogonal to 'angle_labels' (the format). Ignored by the domain scales ("cardinal"/"hours"/"months"/"seconds"), which carry their own count.

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 circumference. When 'TRUE' (default) and no 'label_col' is given, 'radiate()' falls back to the id/group column only if it has at most 12 distinct values, so many-track plots are not buried under per-trial id labels. Set 'label_col' to force labelling regardless of track count.
label_col	Column containing label values. When supplied explicitly, labels are always drawn (the many-track guard applies only to the automatic fallback).
label_size	Text size for circumference 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.
clip_tracks	Logical; when 'TRUE' (default) trajectory paths are clipped to the unit circle, so beyond-circumference ('rho > 1') excursions are truncated at the circumference (segment-intercept, leaving a gap until the track re-enters) rather than drawn past it. Set 'FALSE' to draw tracks unclipped. Plot-only: kinematics are unaffected. 'rho' is unchanged by the relative-frame rotation, so this applies equally when 'coords = "absolute"'. Whether to draw a mean resultant arrow from the centre.
show_arrow	
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 'facets' 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, start_sep, 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")
radiate(cpunctatus, coords = "absolute")
```

read_tracks	<i>Construct a Tracks from a data.frame or file(s)</i>
-------------	--

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 = FALSE,
  origin = NULL,
  radius = NULL,
  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,
  id_collision = c("error", "namespace"),
  on_invalid = c("error", "drop")
)
```

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". Numeric time is taken to be a frame index unless 'time_type = "seconds"' (or the column is a duration): seconds are recorded as such in the Tracks metadata ('time_unit'), so [elapsed_seconds()] and the kinematics use them directly, with no frame rate.
tz	Time zone of timestamps that carry none (default "UTC"): in the time column and the kept columns (see 'keep') of a file, in an Excel date-time cell, and in a text time column of a data frame. A dialect is not given 'tz': it applies to the dates and text date-times of the table the dialect returns. A bare date is the first instant of that day in 'tz' (midnight, or the end of a DST change that skips midnight; so is an Excel date-time cell at such a midnight). A wall-clock time that does not exist in 'tz' (the hour skipped when clocks go forward, or a day skipped altogether) is an error. A timestamp that ends in a zone designator ('Z', '+02:00', '-0130', '+02') is that instant whatever 'tz' is, and a date-time column of a data frame keeps its instants.
fps	Frames-per-second. Required when time_type = "frames" (the time column is kept as a raw frame index). When a single finite positive value is supplied and the time column is frame-indexed, it is stored as the Tracks capture frame rate

	(see [frame_rate()]); [elapsed_seconds()] then performs the frame -> seconds conversion once, on demand, so kinematics and the app adopt it automatically without converting twice.
normalize_xy	FALSE (default) keeps (x,y) as supplied. TRUE independently centres each trajectory on its own bounding-box midpoint and scales it to the unit circle – shape-preserving only, NOT calibration (it does not preserve bearing from a fixed arena origin). See [tracks()] for the full explanation.
origin, radius	Optional fixed unit-circle calibration forwarded to [tracks()]: ‘origin = c(x, y)’ centre and positive scalar ‘radius’ in raw coordinate units. Supplied together, they map every trajectory uniformly via ‘(xy - origin) / radius’, preserving bearings from the origin. Mutually exclusive with ‘normalize_xy = TRUE’.
dialect	optional registered dialect name. The dialect reads ‘x’ itself (‘read_tracks()’ does not parse the file first) and is called once per file when ‘x’ holds several paths. See [register_loader_dialect()].
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) and ‘decimal’ (decimal mark) for a text file, ‘sheet’ (Excel worksheet name or number) for an Excel file, and ‘ext’ (file type, for a path without its extension). Any ‘NULL’ element is auto-detected. Defaults to all-auto. With a ‘dialect’, the non-‘NULL’ entries are handed to the dialect: every built-in takes ‘decimal’; ‘trackmate’, ‘trex’, ‘sleap’ and ‘idtrackerai_wide’ also take ‘delim’ and ‘ext’ for a text file and ‘sheet’ for an Excel file (‘decimal’ and ‘delim’ are refused for Excel, ‘sheet’ for a text file); ‘ethovision’ and ‘anymaze’ take ‘delim’ (for EthoVision it must be the export’s own separator). An entry a dialect cannot use stops with a ‘radiatR_invalid_argument’ error, as does any entry for a registered dialect whose function has no ‘read_opts’ argument. Without a dialect, an entry that does not fit the file’s type (‘delim’ or ‘decimal’ for an Excel or Arrow file, ‘sheet’ for a text or Arrow file) stops the same way. A data frame ‘x’ takes only ‘decimal’, the mark of any numeric column it holds as text.
mutate	list of transformations applied after reading (named functions or formulas)
keep	extra columns to retain besides the mapped ones and those a dialect produces (NULL = none). All other columns are dropped; a name that is not in the data is ignored without a message. A kept column of a file is read as date-times by content, under every reader: when its reader typed it as dates or date-times, or every value that is not missing is a date-time in one layout, it is parsed in ‘tz’ as the time column is (a value that is no instant there stops the load); otherwise it stays as read.
drop	drop these columns after mapping
id_from_filename	if TRUE and id missing, derive id from file stem when reading multiple files
validate	Deprecated and ignored: ‘Tracks’ invariants are always checked. Any value but ‘TRUE’ warns.
format	Optional loader format name or list spec registered via [register_loader_format()]

id_collision	how to handle a trajectory id that appears under more than one source file during a multi-file import: "error" (default) stops with a message naming the colliding id(s) and files; "namespace" rewrites every id as '<file>::<id>', where '<file>' is the file name less extension, or the full file name where two files share that, or the trailing part of the path that tells them apart ('a/n1.csv'). Has no effect on single-file or data.frame input.
on_invalid	How to handle rows whose x/y coordinates are missing or non-finite: "error" (default) aborts with a radiatR_invalid_rows condition listing the affected rows and ids; "drop" removes them and signals a radiatR_dropped_rows warning with the same payload. It covers nothing else: a missing time, id or weight is an error either way, and so is a malformed value (see "Malformed values").

Value

Tracks

Time units from a dialect

A dialect that knows the unit of its time column says so, and under the default 'time_type = "auto"' that decides how the column is read: TrackMate 'POSITION_T' (when the version 7 units row gives seconds), ToxTrac 'Time (sec)', idtracker.ai 'seconds' and TRex 'time' are seconds, and need no 'fps'; 'FRAME'/'frame' columns, the DeepLabCut frame index and SLEAP 'frame_idx' are frames. 'ethovision' ('Trial time', or 'Recording time' through 'time_col') and 'anymaze' ('Time', a duration) declare seconds; an 'ethovision' data frame's time is taken as seconds (EthoVision writes no other unit), and a file's units row is checked. A contrary explicit 'time_type' stops with a 'radiatR_time_type_mismatch' error, as do files of one import that disagree.

Masked points

The 'deeplabcut', 'sleap', 'ethovision' and 'anymaze' dialects take each position from one or more points (bodyparts, nodes, body points; ANY-maze's centre point alone). EthoVision writes '-' and ANY-maze an empty cell for a sample with no position; neither has scores, thresholds, or for 'anymaze' a 'partial' argument. A point is masked in a frame when its coordinates are missing or not finite, when its score (likelihood) is 0, or when a threshold is given ('dialect_args = list(likelihood_min =)' for DeepLabCut, 'score_min' for SLEAP) and its score is missing or below it. A frame with a masked point has no position, unless 'dialect_args = list(partial = "mean")' asks for the score-weighted mean of the points that remain. The dialect never removes a row: frames without a position meet 'on_invalid' like any other missing coordinate.

What was masked is reported in one message (class 'radiatR_attrition') and kept in the 'attrition' element of the Tracks metadata: a data frame with one row per source file and point, giving the rows in which the point was masked ('masked'), the rows read from that file ('frames'), the rows left without a position ('frames_no_position'), and the threshold and 'partial' setting used. 'c()' row-binds these records; a subset keeps the whole record of its load, so recombining subsets of one load repeats it.

Ids

An id is a label, not a measurement. In a delimited file the id column (the one named in 'mapping', or any column named 'id', 'track', 'trajectory', 'animal' or 'subject') is kept as the text in the file

wherever reading it as numbers would merge ids that are written differently: '1.1' and '1.10', '1' and '1.0', '01' and '1', or '2024-01-01 00:00' and '2024-01-01 00:00:00'. Ids that read as distinct numbers (or dates) stay numbers. Where the delimited files of one import do not agree on the type of the id column, or two of them hold one number (written '1.1' in one and '1.10' in the other, or alike), the ids of each file are the text in that file (for an Excel file, the text of its cells); two files that write one id alike collide (see 'id_collision'). Where the types still disagree in tables that cannot be read again for their text (a dialect's tables, Parquet or Feather files), the load stops with a 'radiatR_id_type_mismatch' error. A data frame and an Excel or Arrow file read alone keep the types they come with. Ids that differ only by surrounding whitespace (" 1" and "1 ") stop the load with a 'radiatR_ambiguous_ids' error, as readers disagree on whether they are one id. An 'anymaze' file is read as text, so the column its 'id_col' names keeps its text. For 'ethovision', 'id_col' applies to a data frame only (a file is one track, id "1"). Not covered: an id column that another dialect is pointed to by an argument of its own (its ids may still be merged).

Reading delimited files

Whichever reader is installed (readr, then data.table, then base R), a delimited file is read as one table: "" is the only quote character, nothing is a comment ('#' and "" are text), blank lines are skipped, an empty cell is missing, a cell keeps its surrounding whitespace and column names lose theirs. The fields of each line are counted first: a row with more or fewer fields than the header (a last line cut off, an unclosed quote, a table written with row names or with a separator at the end of every row) stops the load with a 'radiatR_malformed_file' error naming the lines ('problem = "ragged"'), as does a reader returning fewer rows than the file holds ("rows") and a column name used twice ("duplicate_columns"; a dialect handles its format's repeated columns itself), and a delimited file that reads as a single column ("one_column").

Malformed values

A cell in an x, y, angle, weight or time column that is present but does not parse ("oops", "1.5", a stray header or units row) stops the load with a 'radiatR_malformed_values' error, a subclass of 'radiatR_invalid_rows'. Its 'columns', 'rows', 'values', 'source' and 'dialect' fields name the column, the data rows (counted from the first row after the header, within its own file in a multi-file import), the offending strings, the file and the dialect, 'decimal' (the mark the file was read with) and 'suggested_decimal' (the other mark when the values look written with it, else 'NULL'). Such a value is never dropped or turned into 'NA'. Blank cells and 'NA' are missing values; 'nan' and 'inf' are read as such. Files are read with no digit-grouping mark, so '1,5' is malformed unless 'read_opts = list(decimal = ",")' says the comma is the decimal mark. Where it is, a value holding a '.' is malformed in turn, since '1.234' could as well be 1234. A semicolon-separated file is taken to use decimal commas unless its first lines show decimal points and no decimal comma ('1.5' or '0.125', which cannot be digit grouping, or the fraction of a duration such as '0:00:00.320'); 'read_opts = list(decimal = ".")' says so outright. A point found that way must be shown by a column read for a role (x, y, time, angle, weight): where such a column holds only values that could be digit grouping ('523.117', '1.234') and the clear decimal points are all in other columns (a label '1.5'), the load stops with a 'radiatR_ambiguous_decimal' error (fields 'columns', 'values', 'evidence', 'decimal', 'source', 'dialect') rather than read '1.234' for what may be 1234. So does a file where no data row of the first 20 lines is whole (a quoted field runs past them), as the point cannot then be checked. Give the mark with 'read_opts'. The same holds for the built-in dialects that sniff the mark (ANY-maze, the others read as tables, and EthoVision when not comma-separated). Other tab- and comma-separated files are read with a point.

Every built-in dialect checks its source columns the same way, before any reshaping, so the error names the header as the program wrote it and the row of the file's table. A missing or non-finite position found afterwards (see 'on_invalid') is still counted in the rows of the table the dialect returns, which for a wide format is the reshaped one. ANY-maze writes time as a duration ('0:01:02.500'), read as seconds; a cell that is not one is malformed.

A time column given as text must hold either numbers throughout or timestamps in one layout throughout: 'YYYY-MM-DD HH:MM:SS', 'YYYY-MM-DD HH:MM' or 'YYYY-MM-DD' (with '-' or '/'), or ISO 8601 'YYYY-MM-DDTHH:MM:SS' with an optional 'Z' for UTC. A bare date beside date-times is midnight.

Examples

```
df <- data.frame(
  id   = rep(c("A", "B"), each = 4),
  time = rep(0:3, times = 2),
  x     = c(1, 1.2, 1.4, 1.5, -0.2, -0.1, 0, 0.15),
  y     = c(0, 0.1, 0.3, 0.4, 1.0, 1.1, 1.3, 1.5)
)
read_tracks(df, mapping = list(id = "id", time = "time", x = "x", y = "y"))

# A DeeplabCut predictions file, recorded at 40 frames per second
dlc <- system.file("extdata", "dlc_single_wasp_predictions.csv",
  package = "radiatR")
read_tracks(dlc, dialect = "deeplabcut", fps = 40)
```

read_tracks_dir	<i>Read all matching files from a directory and bind into a Tracks</i>
-----------------	--

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()'

Value

A single ‘Tracks’ object combining every matching file.

Examples

```
df <- data.frame(
  id   = rep(c("A", "B"), each = 4),
  time = rep(0:3, times = 2),
  x     = c(1, 1.2, 1.4, 1.5, -0.2, -0.1, 0, 0.15),
  y     = c(0, 0.1, 0.3, 0.4, 1.0, 1.1, 1.3, 1.5)
)
csv_dir <- file.path(tempdir(), "csv_tracks")
dir.create(csv_dir)
write.csv(df, file.path(csv_dir, "day1.csv"), row.names = FALSE)
write.csv(transform(df, id = paste0(id, "2")), file.path(csv_dir, "day2.csv"),
  row.names = FALSE)
read_tracks_dir(csv_dir, pattern = "\\*.csv$")
unlink(csv_dir, recursive = TRUE)
```

read_tracks_format	<i>Construct a Tracks from a *format* spec (registered name or inline list)</i>
--------------------	---

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

Value

A ‘Tracks’ object, as returned by [read_tracks()].

Examples

```
raw <- data.frame(animal = "m1", seconds = c(0, 0.5, 1),
  pos_x = c(0.1, 0.2, 0.3), pos_y = c(0, 0.05, 0.1))
# An inline spec: columns found by pattern, time in seconds
spec <- list(
  regex_mapping = list(id = "^animal", time = "^sec",
```

```

        x = "^pos_x", y = "^pos_y"),
    time = list(type = "seconds")
)
read_tracks_format(raw, spec)

# The same spec registered under a name
register_loader_format("my_tracker", spec, overwrite = TRUE)
read_tracks_format(raw, "my_tracker")

```

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()]

Examples

```

x <- set_reference(cpunctatus[1:3], pi / 2)
reference(x)

```

`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

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

Value

The rule name, invisibly.

See Also

[list_heading_rules](#), [derive_headings](#)

Examples

```
last_bearing <- function(df, cols, ...) {  
  n <- nrow(df)  
  data.frame(id = df[[cols$id]][n], time = df[[cols$time]][n],  
             heading = atan2(df[[cols$y]][n], df[[cols$x]][n]) %% (2 * pi))  
}  
register_heading_rule("last_bearing", last_bearing, overwrite = TRUE)  
head(derive_headings(cpunctatus, rule = "last_bearing"))
```

```
register_loader_dialect
```

Register a custom loader dialect

Description

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

Details

‘x’ is whatever was passed to [read_tracks()]: a data frame, or one file path. The dialect is the only reader of that file, and is called once per file when ‘read_tracks()’ is given several paths. A table returned without an id column is one trajectory, named after its file in a multi-file import. Numeric columns returned as text are coerced strictly (see "Malformed values" in [read_tracks()]) using the decimal mark sniffed from the file; a dialect that knows better can set ‘attr(out, "decimal")’.

Numeric time is read as a frame index unless the dialect says its time column is in seconds, with ‘attr(out, "time_unit") <- "seconds"' (or "'frames"', to rule out ‘time_type = "seconds"'). See "Time units from a dialect" in [read_tracks()].

A dialect that derives each position from several points can report what it left out with ‘attr(out, "attrition")’: a data frame with columns ‘source’, ‘point’, ‘masked’, ‘frames’, ‘frames_no_position’, ‘score_min’ and ‘partial’, one row per point, with whole non-negative counts and one ‘frames’ and one ‘frames_no_position’ value (the per-file totals, repeated on each row). See "Masked points" in [read_tracks()].

A dialect can keep facts about the file it read (a header block, the units of its columns) with ‘attr(out, "source_meta")’: a list with distinct names other than ‘source’ and ‘path’. [read_tracks()] stores one record per file in ‘Tracks@meta\$source_meta’, each the dialect’s list with the file’s name (‘source’) and path (‘path’) in front. The records are a plain list in file order, not named by file, because files in different folders share names.

A loader that reads files can take ‘read_tracks(read_opts =)’ by declaring a ‘read_opts’ argument: it receives the non-‘NULL’ entries (‘decimal’, ‘delim’, ‘sheet’, ‘ext’) and must honour them or stop. A table it returns without ‘attr(, "decimal")’ is coerced with ‘read_opts\$decimal’, else the file’s sniffed mark. Without that argument, any ‘read_opts’ stop the load.

Value

‘name’, invisibly. Called for its side effect of registering the dialect for ‘read_tracks(dialect =)’.

Examples

```
# dtrack writes headerless, tab-separated frame, x, y (and a flag) columns
dtrack_loader <- function(x) {
  d <- utils::read.delim(x, header = FALSE, colClasses = "numeric")
  data.frame(id = tools::file_path_sans_ext(basename(x)),
             frame = d[[1]], x = d[[2]], y = d[[3]])
}
register_loader_dialect("dtrack", dtrack_loader, overwrite = TRUE)
list_loader_dialects()

untar(system.file("extdata", "tracks.tar.gz", package = "radiatR"),
      exdir = tempdir())
track_dir <- file.path(tempdir(), "tracks")
read_tracks(file.path(track_dir, "10_10_point02.txt"), dialect = "dtrack")
```

register_loader_format

*Register a declarative loader *format* (list or YAML/JSON file)*

Description

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

Value

‘name’, invisibly. Called for its side effect of registering the format for [read_tracks_format()].

Examples

```
# A tracker whose columns are found by pattern; time is in seconds
spec <- list(
  regex_mapping = list(id = "^animal", time = "^sec",
    x = "^pos_?x", y = "^pos_?y"),
  time = list(type = "seconds")
)
register_loader_format("my_tracker", spec, overwrite = TRUE)
list_loader_formats()
```

restrict_to_circumference

Restrict a Tracks to within the unit circle

Description

Removes out-of-circumference ($\rho > 1$) track data so that downstream kinematics (`track_speed()`, `instantaneous_speed()`, `track_length()`, the `path_*` shape metrics, ...) are computed only on the within-circle portion of each trajectory. Distance from the centre, ρ , is taken from the relative (unit-circle) coordinates $\text{rel}_x/\text{rel}_y$, where $\rho = 1$ is the circumference.

Usage

```
restrict_to_circumference(ts, mode = c("truncate", "drop"), max_radius = 1)
```

Arguments

<code>ts</code>	A ‘Tracks’ carrying relative coordinates ($\text{rel}_x/\text{rel}_y$).
<code>mode</code>	How to exclude out-of-circumference data per trajectory. “truncate” (default) keeps the contiguous prefix up to (excluding) the first point beyond the circumference – the centre-to-circumference approach segment, with no gaps. “drop” removes individual beyond-circumference points only; note that dropping interior points lengthens the step bridging the gap, which biases speed and path length, so “truncate” is usually preferable for kinematics.
<code>max_radius</code>	Circumference radius. A point is beyond the circumference when $\rho > \text{max_radius}$ (with a small rim tolerance so points exactly on the circumference are kept). Default ‘1’.

Details

This is the data-filtering counterpart to the plot-only `radiate(clip_tracks = TRUE)`: `clip_tracks` only changes what is **drawn** (on a copy), whereas `restrict_to_circumference()` returns a new ‘Tracks’ whose data the metrics actually see.

Value

A new ‘Tracks’ (the input is unmodified) with out-of-circumference rows removed and a `transform_history` entry recording the operation. Trajectories reduced to no rows are dropped.

See Also

[radiate()] ('clip_tracks'), [track_length()], [track_speed()], [transform_history()]

Examples

```
x <- restrict_to_circumference(cpunctatus)
c(before = nrow(as.data.frame(cpunctatus)), after = nrow(as.data.frame(x)))
```

scale_cardinal	<i>Circumference scale: cardinal compass directions</i>
----------------	---

Description

Circumference scale: cardinal compass directions

Usage

```
scale_cardinal(points = 4)
```

Arguments

points '4' (N/E/S/W, default) or '8' (adds the intercardinals NE/SE/SW/NW).

Value

A circumference-scale list for [circumference_labs()].

See Also

[circumference_labs()], [scale_clock()], [scale_months()], [scale_seconds()]

Examples

```
scale_cardinal(points = 8)
```

scale_clock	<i>Circumference scale: clock hours</i>
-------------	---

Description

Draws a tick at every hour and labels a sparse subset.

Usage

```
scale_clock(hours = 24, every = NULL)
```

Arguments

hours	‘24‘ (default) or ‘12‘.
every	Label every ‘every‘ hours. ‘NULL‘ (default) auto-picks ‘6‘ for a 24-hour dial and ‘3‘ for a 12-hour dial. Must evenly divide ‘hours‘.

Value

A circumference-scale list for [circumference_labs()].

See Also

[circumference_labs()], [scale_cardinal()], [scale_months()], [scale_seconds()]

Examples

```
scale_clock(hours = 12)
```

scale_months	<i>Circumference scale: months of the year</i>
--------------	--

Description

Circumference scale: months of the year

Usage

```
scale_months(format = c("abbr", "initial", "number"))
```

Arguments

format	“abbr” (Jan...Dec, default), “initial” (J, F, M, ...), or “number” (1...12).
--------	--

Value

A circumference-scale list for [circumference_labs()].

See Also

[circumference_labs()], [scale_cardinal()], [scale_clock()], [scale_seconds()]

Examples

```
scale_months("initial")
```

scale_seconds

Circumference scale: seconds (or minutes)

Description

Draws 60 ticks and labels a sparse subset. Also serves a minutes dial.

Usage

```
scale_seconds(every = 15)
```

Arguments

every Label every ‘every‘ divisions (default ‘15‘). Must evenly divide 60.

Value

A circumference-scale list for [circumference_labs()].

See Also

[circumference_labs()], [scale_cardinal()], [scale_clock()], [scale_months()]

Examples

```
scale_seconds(every = 10)
```

sector_summary	<i>Proportion of time spent in angular sectors</i>
----------------	--

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 mid-point in radians), count, proportion.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
sector_summary(hd, sectors = 4)
```

set_reference	<i>Set the per-trajectory reference and re-derive the relative frame</i>
---------------	--

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\$rel_y') 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()]

Examples

```
x <- cpunctatus[1:2]
# one reference for every trajectory, in unit-circle radians
reference(set_reference(x, pi / 2))
# or one per trajectory, named by id; a reference of 0 makes the
# relative frame equal the absolute one
x <- set_reference(x, stats::setNames(c(0, pi), ids(x)))
head(as.data.frame(x)[c("trial_id", "abs_theta", "rel_theta")])
```

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 for reproducibility. The simulation is drawn under this seed and the caller's random-number state is restored on exit, so the call neither depends on nor alters the session's RNG stream. With 'NULL' (default) the draws come from the session's stream as usual.
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 $\text{ref_mean} + \text{mean_slope} * \text{predictor}$ 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", "unimodal_uniform" (a directed mode over a uniform background; 'mix_weight' sets the directed fraction), or "bimodal" (an asymmetric two-von-Mises mixture; 'mix_weight', 'mode2_mean', 'kappa2' set the second mode). 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)
```

sinuosity

Per-trajectory sinuosity for a Tracks

Description

Computes `[path_sinuosity()]` (Benhamou 2004) for each trajectory in a ‘Tracks’, ordering each trajectory’s points by its time column when one is recorded. Distance-calibrated: with a ‘distance_scale’ set the step lengths are in real units, so the reported sinuosity is in $1/\sqrt{\text{unit}}$.

Usage

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

Arguments

<code>ts</code>	A 'Tracks'.
<code>x_col, y_col</code>	Names of the coordinate columns to use. Default to the 'Tracks's recorded x/y columns (the real recorded positions).

Value

A 'data.frame' with one row per trajectory: the 'Tracks's id column and a numeric 'sinuosity' column ('>= 0').

See Also

[path_sinuosity()], [straightness_index()], [tortuosity_ratio()], [set_distance_scale()]

Examples

```
head(sinuosity(cpunctatus))
```

stack_headings	<i>Add stacking columns to a headings data frame</i>
----------------	--

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 = step/2,
  tol = NULL,
  direction = "inward",
  base_r = 1,
  shade = FALSE,
  shape = FALSE,
  group = NULL
)
```

Arguments

<code>data</code>	A data frame with an angle column in radians.
<code>col</code>	Name of the angle column. Defaults to the <code>heading_col</code> attribute when <code>data</code> is a <code>headings_frame</code> .
<code>step</code>	Radial gap between successive dots in a stack, in data units (the analogue of <code>circular::plot.circular</code> 's <code>sep</code>). Default <code>0.025</code> matches that package; larger values separate the dots more.
<code>start_sep</code>	Radial offset of the first (outermost, for "inward") dot from <code>base_r</code> , in data units (the analogue of <code>circular::plot.circular</code> 's <code>start_sep</code>). Default <code>step / 2</code> places the first dot half a step inside the reference circle so its outer edge approximately abuts the circumference. Set to <code>0</code> to place the dot centre exactly on the reference circle.
<code>tol</code>	Grouping tolerance in radians. <code>NULL</code> (default) = exact equality, correct for binned data. <code>tol > 0</code> assigns each observation to the nearest group centre within <code>tol</code> radians (greedy, sorted-order scan); angles near <code>0</code> and <code>2*pi</code> are not treated as neighbours.
<code>direction</code>	"inward" (default, stacks toward centre) or "outward" (away from the circumference, matches <code>circular</code> default).
<code>base_r</code>	Radius of the reference circle in data units. Default <code>1</code> .
<code>shade</code>	If <code>TRUE</code> , add a <code>shade_n</code> column (alias of <code>stack_n</code>) for use as an alpha aesthetic.
<code>shape</code>	If <code>TRUE</code> , add a <code>shape_code</code> integer column: <code>1</code> = hollow (outermost / singleton), <code>2</code> = filled (middle), <code>3</code> = filled with ring (innermost in a stack of 3+).
<code>group</code>	Optional column name; when set, stacking is computed independently within each group and the rows recombined. Default <code>NULL</code> .

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](#)

Examples

```
d <- data.frame(heading = bin_angles(c(0.10, 0.12, 0.13, 1.50, 1.52),
                                     width = pi / 18))
stack_headings(d, col = "heading")
```

step_speed	<i>Per-step speed along a trajectory</i>
------------	--

Description

Speed of each step as straight-line step distance divided by the elapsed time of that step: $\sqrt{\text{diff}(x)^2 + \text{diff}(y)^2} / \text{diff}(\text{seconds})$. The unit is the distance unit of 'x'/'y' per second; for radiatR's unit-circle coordinates that is 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 $\text{length}(x) - 1$. 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	<i>Per-trajectory straightness index for a Tracks</i>
--------------------	---

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()]

Examples

```
head(straightness_index(cpunctatus))
```

test_concentration	<i>Test whether groups share the same concentration (dispersion)</i>
--------------------	--

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 <code>equal.kappa.test</code> ; FALSE uses <code>wallraff.test</code> .
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.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "type")
test_concentration(hd, group_col = "type")
```

test_distributions	<i>Test whether groups share the same circular distribution</i>
--------------------	---

Description

Compares the angular distributions of two or more groups. Complements [test_mean_directions](#) (which tests only the mean direction) and [test_concentration](#) (which tests only the spread) by testing the distributions as a whole, or – for method = "rao" – the mean and the dispersion jointly.

Usage

```
test_distributions(
  hd,
  group_col,
  angle_col = "heading",
  method = c("watson_wheeler", "watson_two", "rao"),
  pairwise = FALSE,
  p_adjust = "none",
  axial = FALSE,
  p_method = c("table", "monte_carlo"),
  n_perm = 9999L
)
```

Arguments

hd	Data frame with heading and group columns.
group_col	Column identifying the groups to compare.
angle_col	Heading column in radians. Default "heading".
method	One of "watson_wheeler" (default), "watson_two", or "rao".
pairwise	Logical. FALSE (default) runs a single omnibus test across all groups ("watson_two" then requires exactly two groups). TRUE runs every pairwise comparison.
p_adjust	Multiple-comparison correction passed to p.adjust . Default "none". Applies only to the pairwise output, where a p_value_adj column is added; "BH" or "holm" are recommended. Meaningless for the tabled "watson_two" p-values.
axial	Logical. Treat the angles as axial (bidirectional, mod-pi) data: the test is run via the angle-doubling method. Default FALSE.
p_method	For method = "watson_two" only: "table" (default) reports the tabled significance band; "monte_carlo" instead returns a continuous permutation p-value from n_perm label shuffles. Ignored by the other methods (which already give continuous p-values).
n_perm	Number of label permutations for p_method = "monte_carlo". Default 9999. Set the RNG seed with set.seed for reproducible p-values.

Details

"watson_wheeler" (**default**) Mardia-Watson-Wheeler uniform-scores test: a non-parametric k-sample test that makes no von Mises assumption. Returns an exact (chi-squared) p-value.

"watson_two" Watson's two-sample U^2 test of homogeneity. Strictly two-sample: with more than two groups set pairwise = TRUE. The circular package returns only tabled critical values, so by default p_value is the upper edge of a tabled band (e.g. 0.05 means $0.01 < p < 0.05$) and is NA when $p > 0.10$. Set p_method = "monte_carlo" for a continuous permutation p-value.

"rao" Rao's test for homogeneity of both mean directions and dispersions across k groups; returns two rows per comparison, one for each component ("mean", "dispersion").

Value

A tidy data frame. Omnibus output has columns method, component, statistic, df, p_value, n_groups, n. Pairwise output replaces n_groups with group1/group2 and adds p_value_adj when p_adjust != "none".

See Also

[test_mean_directions](#), [test_concentration](#), [test_uniformity](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "type")
test_distributions(hd, group_col = "type")
```

test_gof

*Goodness-of-fit test against a wrapped Cauchy distribution***Description**

Tests whether each group's headings plausibly come from a wrapped Cauchy distribution, using Watson's U^2 statistic on the probability-integral transform of the fitted distribution, with a parametric-bootstrap p-value (the bootstrap accounts for the mean direction and concentration having been estimated from the same sample, which invalidates the naive tabled p-value for `watson.test()`). Unlike `test_uniformity`, which tests "is this uniform?", `test_gof` tests a different null hypothesis: "does this fit a wrapped Cauchy?" — useful when a unimodal but non-von-Mises shape (heavier or lighter tails) is suspected.

Usage

```
test_gof(
  hd,
  group_col = NULL,
  angle_col = "heading",
  dist = c("wrappedcauchy"),
  p_adjust = "none",
  n_boot = 999L
)
```

Arguments

<code>hd</code>	Data frame with a heading column in radians.
<code>group_col</code>	Column to group by. <code>NULL</code> tests the whole data frame as one group.
<code>angle_col</code>	Heading column name. Default "heading".
<code>dist</code>	Candidate distribution family. Currently only "wrappedcauchy" is implemented.
<code>p_adjust</code>	Multiple-comparison correction method passed to <code>p.adjust</code> . Default "none". Applies only when <code>group_col</code> is supplied; a <code>p_value_adj</code> column is added to the result.
<code>n_boot</code>	Number of parametric bootstrap replicates for the p-value. Default 999. Each replicate refits the wrapped-Cauchy MLE, so this is more expensive per-replicate than the package's other Monte-Carlo p-values; lower it for quick exploratory calls. Set the RNG seed with <code>set.seed</code> for reproducible p-values.

Value

Tidy data frame with columns `group_col` (if supplied), `statistic`, `p_value`, `n`, `mu` (fitted mean direction, radians), `rho` (fitted concentration, $[\emptyset, 1)$), `test`, and `p_value_adj` (when `p_adjust != "none"`). Returns `NULL` (ungrouped) or omits a group's row (grouped) when that sample has fewer than 3 finite angles or the fit fails to converge. If the wrapped-Cauchy fit succeeds but the bootstrap step itself fails, the row is kept with valid `statistic/mu/rho` but `p_value` set to `NA`.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
set.seed(1)
# n_boot kept small to run quickly; use the default in an analysis
test_gof(hd[hd$arc == "50", ], n_boot = 99)
```

test_mean_directions *Test whether groups share the same mean direction*

Description

Two tests of equal mean direction across groups:

method = "watson_williams" (**default**) `watson.williams.test` – the circular analogue of the parametric F -test for equal means. Assumes von Mises-distributed data with equal concentrations across groups.

method = "permutation" A distribution-free permutation test using a between-/within-group resultant-length statistic, with the p-value obtained by shuffling group labels. Makes no von Mises assumption and is robust when Watson-Williams' F calibration is doubtful (moderate concentration, non-von-Mises shape). Note it assumes exchangeability under the null, so like Watson-Williams it is most trustworthy when group concentrations are similar.

Usage

```
test_mean_directions(
  hd,
  group_col,
  angle_col = "heading",
  method = c("watson_williams", "permutation"),
  pairwise = FALSE,
  p_adjust = "none",
  axial = FALSE,
  n_perm = 9999L
)
```

Arguments

hd	Data frame with heading and group columns.
group_col	Column identifying conditions or groups.
angle_col	Heading column in radians. Default "heading".
method	One of "watson_williams" (default) or "permutation".
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).
n_perm	Number of label permutations for method = "permutation". Default 9999. Ignored otherwise. Set the RNG seed with set.seed for reproducible p-values.

Value

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

See Also

[test_distributions](#), [test_concentration](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
test_mean_directions(hd, group_col = "arc")
set.seed(1)
test_mean_directions(hd, group_col = "arc", method = "permutation",
                     n_perm = 199)
```

test_symmetry	<i>Test a circular distribution for reflective symmetry</i>
---------------	---

Description

Pewsey's (2002) omnibus test of reflective symmetry about an unspecified mean direction, based on the standardised second central sine moment. The null hypothesis is that the distribution is symmetric about its mean direction; a small p-value indicates skewness (asymmetry). Useful as a precondition check before applying methods that assume symmetry, such as a von Mises fit or the Watson-Williams test.

Usage

```
test_symmetry(
  hd,
  group_col = NULL,
  angle_col = "heading",
  p_adjust = "none",
  axial = FALSE
)
```

Arguments

hd	Data frame with a heading column in radians.
group_col	Column to group by. NULL (default) tests the whole data frame as one group.
angle_col	Heading column name. Default "heading".
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.
axial	Logical. Treat the angles as axial (bidirectional, mod- π) data: symmetry is tested via the angle-doubling method. Default FALSE.

Details

The test uses a large-sample normal approximation and can be liberal at small n or low concentration; treat borderline results at small samples with caution.

Value

Tidy data frame with columns group_col (if supplied), statistic (the standardised $|\bar{b}_2|$), p_value, n, test, and p_value_adj (when p_adjust != "none"). Groups with fewer than four finite angles yield NA statistics.

References

Pewsey, A. (2002). Testing circular symmetry. *Canadian Journal of Statistics* 30(4), 591–600. [doi:10.2307/3316098](#).

See Also

[test_uniformity](#), [vonmises_fit](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
test_symmetry(hd, group_col = "arc")
```

test_uniformity	<i>Per-group tests of circular uniformity</i>
-----------------	---

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, so by default the p_value column contains the tabled significance level rather than a continuous p-value. Set p_method = "monte_carlo" to obtain a continuous p-value for those three by simulation instead (usable with p_adjust).

Usage

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

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), "vtest", "kuiper", "rao", "watson", "hermans_rasson", or "pycke". "vtest" is the V-test: uniformity against a <i>specified</i> mean direction mu, considerably more powerful than the Rayleigh test when a bearing is expected a priori. The Hermans-Rasson and Pycke omnibus tests are far more powerful than Rayleigh against multimodal / non-symmetric alternatives; both obtain p_value 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. Meaningful only for tests with continuous p-values: Rayleigh, the V-test, Hermans-Rasson, Pycke, and any test run with p_method = "monte_carlo".
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" and "pycke" p-values and for p_method = "monte_carlo". Default 9999. Set the RNG seed with set.seed for reproducible p-values.
mu	Numeric scalar, radians. The expected mean direction for the V-test (test = "vtest"); required for that test and ignored otherwise.
p_method	For "kuiper", "rao", and "watson" only: "table" (default) reports the tabled significance level from the circular package (or NA when the statistic falls outside the table); "monte_carlo" instead simulates n_sim uniform samples to obtain a continuous p-value. Ignored by the other tests.

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.

Pycke, J.-R. (2010). Some tests for uniformity of circular distributions powerful against multimodal alternatives. *Canadian Journal of Statistics*, 38(1), 80-96. doi:10.1002/cjs.10048.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
test_uniformity(hd, group_col = "arc", p_adjust = "BH")
# the target lies at 0 rad: test for orientation towards it
test_uniformity(hd, group_col = "arc", test = "vtest", mu = 0)
```

test_unimodality	<i>Test a circular sample for unimodality against bimodality</i>
------------------	--

Description

A parametric-bootstrap likelihood-ratio test of a single von Mises distribution (unimodal) against an asymmetric two-component von Mises mixture (bimodal). The null hypothesis is unimodality; a small p-value is evidence of a second mode.

Usage

```
test_unimodality(hd, group_col = NULL, angle_col = "heading", n_boot = 199L)
```

Arguments

hd	Data frame with a heading column in radians.
group_col	Column to group by. NULL (default) tests the whole data frame as one group.
angle_col	Heading column name. Default "heading".
n_boot	Number of parametric-bootstrap replicates. Default 199.

Details

The mixture likelihood-ratio statistic has a non-standard null distribution (a boundary / non-identifiability problem), so its p-value cannot be read from a χ^2 . Instead it is obtained by a parametric bootstrap: `n_boot` samples are simulated from the fitted von Mises, both models are refitted to each, and the observed statistic is compared with the resulting null distribution. This is deliberately *not* a critical-bandwidth (Silverman) test: the critical bandwidth's null distribution is not valid for circular data (Fisher & Marron 2001), whereas this bootstrap is well calibrated.

The test is computationally intensive – every bootstrap replicate refits the mixture – so it is best run with a modest `n_boot`. Set the RNG seed with `set.seed` for reproducible p-values. It reuses the fitters behind `circ_model_select`, which offers a complementary information-criterion view of the same unimodal / bimodal comparison.

Value

Tidy data frame with columns `group_col` (if supplied), `statistic` (the likelihood-ratio statistic), `p_value`, `n`, and `test`. Groups with fewer than ten finite angles, or where the von Mises fit fails, yield NA statistics.

References

Silverman, B.W. (1981). Using kernel density estimates to investigate multimodality. *JRSS B* 43(1), 97–99. Fisher, N.I. & Marron, J.S. (2001). Mode testing via the excess mass estimate. *Biometrika* 88(2), 499–517.

See Also

[circ_model_select](#), [test_symmetry](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
set.seed(1)
# n_boot kept small to run quickly; use the default in an analysis
test_unimodality(hd[hd$arc == "20", ], n_boot = 19)
```

tortuosity_ratio	<i>Per-trajectory tortuosity ratio for a Tracks</i>
------------------	---

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()]

Examples

```
head(tortuosity_ratio(cpunctatus))
```

track_duration	<i>Duration of each track</i>
----------------	-------------------------------

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()]

Examples

```
head(track_duration(cpunctatus, units = "minutes"))
```

track_length	<i>Per-trajectory path length for a Tracks</i>
--------------	--

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()]

Examples

```
# unit-circle coordinates, so lengths are in arena radii
head(track_length(cpunctatus))
```

track_speed	<i>Per-trajectory speed for a Tracks, in real units</i>
-------------	---

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 radii per second, because radiatR normalises trajectories to a unit circle.

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()]

Examples

```
frame_rate(cpunctatus) # one frame every 5 s
head(track_speed(cpunctatus))
head(track_speed(cpunctatus, stat = "max"))
```

track_turning	<i>Per-trajectory turning-rate summary for a Tracks</i>
---------------	---

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()]

Examples

```
head(track_turning(cpunctatus, units = "degrees"))
```

track_velocity	<i>Per-trajectory net velocity for a Tracks</i>
----------------	---

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()]

Examples

```
tv <- track_velocity(cpunctatus)
head(tv)
# net direction of travel of each track
head(atan2(tv$vy, tv$vx))
```

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 = FALSE,
  origin = NULL,
  radius = NULL,
  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

<code>df</code>	data.frame in long form
<code>id, time</code>	Columns naming trajectory id and time
<code>angle</code>	Optional angle column (radians or degrees, see <code>angle_unit</code>)
<code>x, y</code>	Optional cartesian columns; if provided, converted to unit circle and angle inferred
<code>rel_x, rel_y</code>	Optional column names for pre-transformed relative coordinates (centred on the origin, unit-circle scaled). Both must be supplied together or not at all.
<code>angle_unit</code>	Units of provided angle ("radians" or "degrees"); stored internally as radians
<code>weight</code>	Optional weight column name; weights must be finite and ≥ 0
<code>normalize_xy</code>	If FALSE (default), (x,y) are kept as supplied. If TRUE, each trajectory is independently centred on its own bounding-box midpoint and scaled so its furthest point sits at radius 1. This is a shape-preserving transform ONLY: because each trajectory is translated by a different amount, it does NOT preserve bearing relative to a fixed arena origin, and must not be used as a substitute for metric calibration. Raw coordinates are retained in <code><x>_raw</code> / <code><y>_raw</code> when enabled. Prefer pre-calibrated input, or the landmark-relative frame (<code>'rel_x'</code> / <code>'rel_y'</code> , see <code>'coords = "relative"'</code> in <code>[derive_headings()]</code>) when a per-trial reference direction is available.
<code>origin</code>	Optional length-2 numeric <code>'c(x, y)'</code> giving a fixed unit-circle centre in the raw coordinate units. Supplied together with <code>'radius'</code> , it calibrates every trajectory uniformly via <code>'(xy - origin) / radius'</code> , a similarity that preserves bearings measured from the origin (unlike <code>'normalize_xy'</code> , which centres each trajectory on its own bounding box). Mutually exclusive with <code>'normalize_xy = TRUE'</code> . Raw coordinates are retained in <code><x>_raw</code> / <code><y>_raw</code> . For example, in an experimental arena <code>'origin'</code> is the arena centre and <code>'radius'</code> its edge distance in pixels.
<code>radius</code>	Optional positive finite scalar: the raw distance mapping to <code>'rho = 1'</code> . Required together with <code>'origin'</code> .
<code>meta</code>	Free-form list of metadata
<code>transform_history</code>	Optional tibble describing transformation steps applied to the trajectories. Must contain columns <code>'step'</code> , <code>'order'</code> , <code>'id'</code> , <code>'implementation'</code> , <code>'params'</code> , and <code>'depends_on'</code> .
<code>i</code>	Trajectory identifiers (character ids, numeric indices, or logical vector)
<code>...</code>	Additional Tracks objects to append. All inputs must share identical column mapping and angle unit, have no colliding trajectory ids, and agree on <code>frame_rate</code> / <code>time_unit</code> / <code>distance_scale</code> metadata (including all-unset); mismatches error rather than silently picking one input's values. Per-id metadata (e.g. <code>'reference'</code>) and the attrition records of the loads are merged across all inputs. A subset carries the whole record of its load, so recombining subsets of one load repeats that record.
<code>recursive</code>	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

Invariants

Every Tracks satisfies the following. They are checked when the object is built and whenever `methods::validObject()` runs, which the package's own transforms do (a direct slot write is not checked until then):

- ids are never missing, and the rows of each id are contiguous;
- time is finite and strictly increasing within each id, so no time is repeated within a trajectory;
- angles are finite and lie in $[0, 2\pi)$;
- coordinates are finite: x/y, and the relative and raw pairs where those columns are stored;
- weights, when a weight column is named, are finite and ≥ 0 (a zero weight excludes an observation).

A Tracks therefore never holds a missing detection: remove those before construction (see `on_invalid` in [read_tracks](#)).

`tracks()` reports violations as an error of class `radiatR_invalid_tracks`. Its `problems` field is a data frame with one row per rule, column and row: rule, column, row (the row of the data frame supplied) and id; rows and ids give the union. When called through [read_tracks](#), rows are those of the combined table passed to `tracks()` (after any rows removed by `on_invalid = "drop"`), not line numbers in a file.

Examples

```
df <- data.frame(
  id   = rep(c("a", "b"), each = 3),
  time = rep(1:3, 2),
  x     = c(0, 0.3, 0.6, 0, -0.2, -0.5),
  y     = c(0, 0.2, 0.5, 0, 0.3, 0.7)
)
ts <- tracks(df, x = "x", y = "y")
ts
length(ts)
ts["b"]

# combine objects with the same column mapping
more <- tracks(data.frame(id = "c", time = 1:2, x = c(0, 0.4), y = c(0, -0.4)),
               x = "x", y = "y")
length(c(ts, more))

# pixel coordinates: arena centre (320, 240), edge 200 px from it
px <- data.frame(id = "d", time = 1:3,
                 x = c(320, 380, 450), y = c(240, 280, 330))
tracks(px, x = "x", y = "y", origin = c(320, 240), radius = 200)
```

transform_history	<i>Transform history helpers for Tracks objects</i>
-------------------	---

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.

Examples

```
df <- data.frame(id = "a", time = 1:3, x = c(0, 0.3, 0.6), y = c(0, 0.2, 0.5))
ts <- tracks(df, x = "x", y = "y")
ts <- log_transform(ts, "smooth", params = list(list(window = 3)))
transform_history(ts)
# replace the whole log, here with an empty one
ts <- set_transform_history(ts, transform_history(ts)[0, ])
nrow(transform_history(ts))
```

velocity_angle	<i>Per-row movement direction</i>
----------------	-----------------------------------

Description

The heading of the velocity vector at each observation – the direction of travel from the previous point – as a per-row vector aligned to the Tracks' rows. The directional sibling of [instantaneous_speed()]; the per-observation counterpart of a velocity-based [derive_headings()] rule (which gives one heading per track).

Usage

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

Arguments

ts	A [Tracks-class] object.
units	"radians" (default, '[0, 2*pi)') or "degrees" ('[0, 360)').
x_col, y_col	Position columns. Default the Tracks' 'x'/'y'.

Details

Returned in the package's heading convention: radians in `'[0, 2*pi]'` (or degrees in `'[0, 360]'`), `'0'` = the positive x-axis, increasing counterclockwise – the same frame as `[derive_headings()]` output, so it drops straight into `[circ_summary()]` / `[radiate()]`. Map to a display orientation with `[circ_display()]` when plotting.

Direction needs a step, so the first row of each track is `'NA'`. The result is invariant to any `[set_distance_scale()]` (the scale cancels in the ratio).

Value

A numeric vector, one element per row of `'ts'`, `'NA'` at each track's first row.

See Also

`[velocity_vector()]`, `[instantaneous_speed()]`, `[angular_velocity()]`, `[circ_summary()]`

Examples

```
ts <- simulate_tracks(conditions = data.frame(n_trials = 3L),
                    n_points = 20, seed = 1, output = "trajset")
ts <- set_frame_rate(ts, 30)
head(velocity_angle(ts))
```

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

<code>ts</code>	A <code>'Tracks'</code> .
<code>x_col, y_col</code>	Names of the coordinate columns. Default to the <code>'Tracks'</code> '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()]

Examples

```
head(velocity_vector(cpunctatus))
```

vonmises_fit	<i>Fit a von Mises distribution to per-group heading data</i>
--------------	---

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- π) 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`.

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
vonmises_fit(hd, group_col = "arc")
```

wrappedcauchy_fit	<i>Fit a wrapped Cauchy distribution to per-group heading data</i>
-------------------	--

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

<code>hd</code>	Data frame containing headings in radians.
<code>group_col</code>	Column(s) to group by. <code>NULL</code> fits a single model.
<code>angle_col</code>	Name of the heading column. Default "heading".
<code>axial</code>	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 the convergence flag (0 = the fixed-point MLE iteration converged; 1 = it did not, so treat that row's estimates with caution). 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, 1 = not converged; NA when the fit could not be attempted), n.

See Also

[vonmises_fit](#), [add_wrappedcauchy_density](#)

Examples

```
hd <- derive_headings(cpunctatus, rule = "distal", carry = "arc")
wrappedcauchy_fit(hd, group_col = "arc")
```

zone_dwell	<i>Dwell-time proportions across quadrant x ring zones</i>
------------	--

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

```
# Treat 45 degrees (NE) as the target, as for a water maze probe trial
data(cpunctatus)
dwell <- zone_dwell(cpunctatus, target_angle = pi / 4,
                    ring_breaks = c(0, 0.5, 0.8, 1))
# Q1 proportion > 0.25 indicates above-chance target preference
head(dwell)
```

Index

- * **datasets**
 - cpunctatus, 67
 - [,Tracks,ANY,missing,missing-method (Tracks-class), 144
- add_angle_rose, 5, 14, 35, 99
- add_circ, 7, 17
- add_circ_interval, 8
- add_circ_mean, 9
- add_circular_boxplot, 10
- add_circular_density, 12
- add_circular_kde, 14, 38
- add_critical_r, 16, 17, 19
- add_critical_v_line, 17
- add_heading_arrow, 16, 17, 19, 19
- add_heading_density, 20
- add_heading_interval, 22
- add_heading_points, 24, 33, 99
- add_heading_vectors, 25
- add_landmark, 26
- add_multiple_circles, 28
- add_origin_point, 29
- add_quadrant_lines, 29
- add_radial_grid, 30
- add_stacked_headings, 31, 44, 107
- add_stimulus_arc, 33
- add_ticks, 34
- add_vonmises_density, 14, 35, 37, 38, 151
- add_wrappedcauchy_density, 36, 151, 152
- angular_velocity, 38
- apply_transform, 39
- as.data.frame.Tracks, 40
- as_angle, 41
- assign_color_key (assign_colour_key), 42
- assign_colour_key, 42
- assign_cycle_colors (assign_cycle_colours), 43
- assign_cycle_colours, 43
- bin_angles, 44
- boot_kappa_ci, 45, 47, 57, 65
- boot_kappa_contrast, 46, 46
- boot_mean_ci, 46, 47
- bw.nrd.circular, 15
- c,Tracks-method (Tracks-class), 144
- calibrate_distance (distance_scale), 74
- circ_boxplot_stats, 48
- circ_cor, 49, 55
- circ_dispersion, 46, 48, 51, 99, 150
- circ_display, 51, 56
- circ_model_select, 52, 126, 138, 139
- circ_regression, 54, 64, 65
- circ_summarise, 55
- circ_summary, 56, 56, 58
- circ_summary,Tracks-method (circ_summary), 56
- circ_summary_headings, 57
- circular_mapping, 58
- circumference_labs, 59
- compute_circ_interval, 60
- compute_circ_mean, 61
- compute_circular_density, 62
- concentration_regression, 64
- cor.circular, 50
- count_goal_entries, 66
- cpunctatus, 67
- cycle_colors (cycle_colours), 68
- cycle_colours, 68
- degree_labs, 69
- density.circular, 14
- derive_coords, 70
- derive_headings, 71, 115, 126
- derive_headings,Tracks-method (derive_headings), 71
- directedness_arrow, 73
- distance_scale, 74
- distance_scale,Tracks-method (distance_scale), 74

- distance_unit(distance_scale), 74
- distance_unit, Tracks-method
 - (distance_scale), 74
- dwrappedcauchy, 37
- elapsed_seconds, 75
- fitted.circ_regression
 - (circ_regression), 54
- fitted.concentration_regression
 - (concentration_regression), 64
- fitted_directions, 76
- frame_rate, 77
- frame_rate, Tracks-method (frame_rate), 77
- get_all_object_pos, 78
- get_tracked_object_pos, 79
- get_trial_limits, 81
- gg_traj, 82
- gg_traj, Tracks-method (gg_traj), 82
- guess_columns, 83
- headings_frame, 32, 33, 84, 107, 128
- hf_accessors, 85
- hf_color_col (hf_accessors), 85
- hf_colour_col (hf_accessors), 85
- hf_coords (hf_accessors), 85
- hf_display (hf_accessors), 85
- hf_heading_col (hf_accessors), 85
- ids, 86
- ids, Tracks-method (ids), 86
- import_info, 86
- import_tracks, 87
- instantaneous_speed, 88
- launch_app, 89
- length, Tracks-method (Tracks-class), 144
- line_circle_intercept, 90
- list_heading_rules, 90, 115
- list_loader_dialects, 91
- list_loader_formats, 91
- lm.circular, 54
- load_manifest, 92
- log_transform(transform_history), 147
- log_transform, Tracks-method
 - (transform_history), 147
- optim, 64
- p.adjust, 132, 133, 135–137
- path_sinusosity, 93
- path_straightness, 94
- path_tortuosity, 95
- plot_profile, 96
- plot_speed_direction, 97
- plot_speed_histogram, 98
- pose_to_headings, 6, 51, 99
- predict.circ_regression
 - (circ_regression), 54
- predict.concentration_regression
 - (concentration_regression), 64
- print.circ_regression
 - (circ_regression), 54
- print.concentration_regression
 - (concentration_regression), 64
- rad_shepherd, 100
- radial_theme, 101
- radiate, 5, 14, 35, 37, 84, 101
- read_tracks, 107, 146
- read_tracks_dir, 112
- read_tracks_format, 113
- reference, 114
- reference, Tracks-method (reference), 114
- register_heading_rule, 90, 115
- register_loader_dialect, 116
- register_loader_format, 117
- restrict_to_circumference, 118
- runApp, 89
- scale_cardinal, 119
- scale_clock, 120
- scale_months, 120
- scale_seconds, 121
- sector_summary, 99, 122
- set.seed, 45, 47, 48, 132, 133, 135, 137, 138
- set_distance_scale(distance_scale), 74
- set_distance_scale, Tracks-method
 - (distance_scale), 74
- set_frame_rate(frame_rate), 77
- set_frame_rate, Tracks-method
 - (frame_rate), 77
- set_reference, 122
- set_reference, Tracks-method
 - (set_reference), 122
- set_transform_history
 - (transform_history), 147

set_transform_history, Tracks-method
 (transform_history), 147
simulate_tracks, 55, 64, 65, 123
sinuosity, 126
stack_headings, 31–33, 44, 127
step_speed, 129
straightness_index, 129
summary.circ_regression
 (circ_regression), 54
summary.concentration_regression
 (concentration_regression), 64

test_concentration, 46, 47, 65, 130, 131,
 132, 135
test_distributions, 131, 135
test_gof, 133
test_mean_directions, 131, 132, 134
test_symmetry, 135, 139
test_uniformity, 53, 126, 132, 133, 136, 136
test_unimodality, 138
tortuosity_ratio, 139
track_duration, 140
track_length, 141
track_speed, 141
track_turning, 142
track_velocity, 143
tracks (Tracks-class), 144
Tracks-class, 144
transform_history, 147
transform_history, Tracks-method
 (transform_history), 147

velocity_angle, 148
velocity_vector, 149
vonmises_fit, 36, 45–48, 53, 55, 136, 150,
 151, 152

watson.williams.test, 134
wrappedcauchy_fit, 37, 151

zone_dwell, 152