Package 'tectonicr'

November 3, 2026

Title Analyzing the Orientation of Maximum Horizontal Stress

Version 0.4.10

Description Models the direction of the maximum horizontal stress using relative plate motion parameters. Statistical algorithms to evaluate the modeling results compared with the observed data. Provides plots to visualize the results. Methods described in Stephan et al. (2023) <doi:10.1038/s41598-023-42433-2> and Wdowinski (1998) <doi:10.1016/S0079-1946(98)00091-3>.

License GPL (>= 3)

URL https://tobiste.github.io/tectonicr/

BugReports https://github.com/tobiste/tectonicr/issues

Depends R (>= 4.1.0)

Imports boot, circular (>= 0.5.0), dplyr, ggplot2, lifecycle, methods, RColorBrewer, sf, smoothr (>= 1.0.1), spatstat.explore (>= 3.2.7), spatstat.utils (>= 3.0.4), terra, tidyr, viridis, zoo (>= 1.8.12)

Suggests ggforce, grid, knitr, rmarkdown, roxygen2, testthat (>= 3.0.0), tidyterra

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Tobias Stephan [aut, cre] (ORCID: <https://orcid.org/0000-0002-9290-014X>)

Maintainer Tobias Stephan <tobias.stephan1@yahoo.com>

Repository CRAN

Date/Publication 2026-11-03 11:10:02 UTC

Contents

abs_vel . . . 4
angle-conversion . . . 5
angle_vectors . . . 5
ar_test . . . 6
axes . . . 7
circle_mean_diff . . . 8
circle_stats . . . 9
circular_density . . . 11
circular_dispersion_boot . . . 13
circular_mode . . . 14
circular_plot . . . 15
circular_qqplot . . . 16
circular_range . . . 17
circular_sd_error . . . 18
circular_summary . . . 19
compact-grid . . . 20
confidence . . . 21
confidence_interval_fisher . . . 23
coordinates . . . 24
coordinates2 . . . 25
coordinate_mod . . . 26
cpm_models . . . 26
cunif . . . 28
data2PoR . . . 29
deviation_norm . . . 29
deviation_shmax . . . 30
dispersion . . . 31
distance_binned_stats . . . 33
distance_from_pb . . . 34
dist_greatcircle . . . 36
earth_radius . . . 37
equal_kappa_test . . . 37
equivalent_rotation . . . 39
estimate-kappa . . . 40
euler_pole . . . 41
geom_azimuth . . . 42
geom_azimuthpoint . . . 43
get_azimuth . . . 45
hodges_ajne_test . . . 46
homing . . . 47
import_WSM . . . 48
is.euler . . . 50
kuiper_test . . . 51
line_azimuth . . . 52
mean_resultant_length . . . 53
model_shmax . . . 54

norm_chisq . . . 55
nuvel1 . . . 57
nuvel1_plates . . . 57
ort-eigen . . . 58
ortensor2d . . . 59
parse_wsm . . . 61
pb2002 . . . 61
plates . . . 62
plot_density . . . 63
plot_points . . . 65
PoR2Geo_azimuth . . . 67
PoR_azi . . . 68
PoR_coordinates . . . 69
PoR_crs . . . 70
PoR_distance . . . 71
PoR_map . . . 72
PoR_stress2grid . . . 73
por_transformation . . . 75
prd_err . . . 76
projected_pb_strike . . . 77
quick_plot . . . 78
rao_spacing_test . . . 79
rayleigh-test . . . 80
relative_rotation . . . 83
rolling_test . . . 84
roll_circstats . . . 87
rose . . . 88
rose_geom . . . 90
rose_stats . . . 91
sample_dispersion . . . 93
sample_median . . . 94
second_central_moment . . . 95
shortest_distance_to_line . . . 96
spec_atan . . . 96
spherical_angle . . . 97
stress2grid . . . 98
stress_analysis . . . 102
stress_colors . . . 103
stress_data . . . 103
stress_paths . . . 105
striae . . . 106
superimposed_shmax . . . 107
superimposed_shmax_PB . . . 108
tectonicr.colors . . . 109
uniform_scores_test . . . 110
vcross . . . 111
vm_qqplot . . . 111
vonmises . . . 113

walraff_test 114
watson_test 116
watson_two_sample 118
watson_wheeler_test_perm 120
watson_williams_test 121
wcauchy 123
weighted-rayleigh-test 124
weighting 126
wnorm 127

Index **129**

`abs_vel` *Absolute Plate Velocity*

Description

Calculates the absolute angular velocity of plate motion

Usage

```
abs_vel(w, alpha, r = earth_radius())
```

Arguments

`w`	Angular velocity or rate or angle of rotation
`alpha`	Angular distance to Euler pole or small circle around Euler pole
`r`	Radius. Default is WGS84 Earth's radius (6371.009 km)

Value

numeric (unit of velocity: km/Myr)

See Also

`earth_radius()`

Examples

```
abs_vel(0.21, 0)
abs_vel(0.21, 45)
abs_vel(0.21, 90)
```

angle-conversion *Degrees to Radians*

Description

Helper functions to transform between angles in degrees and radians.

Usage

```
rad2deg(rad)

deg2rad(deg)
```

Arguments

rad	(array of) angles in radians.
deg	(array of) angles in degrees.

Value

numeric. angle in degrees or radians.

Examples

```
deg2rad(seq(-90, 90, 15))
rad2deg(seq(-pi / 2, pi / 2, length = 13))
```

angle_vectors *Angle Between Two Vectors*

Description

Calculates the angle between two vectors

Usage

```
angle_vectors(x, y)
```

Arguments

x, y	Vectors in Cartesian coordinates. Can be vectors of three numbers or a matrix of 3 columns (x, y, z)

Value

numeric. angle in degrees

Examples

```
u <- c(1, -2, 3)
v <- c(-2, 1, 1)
angle_vectors(u, v) # 96.26395
```

`ar_test` *Angular Randomisation Test of Homogeneity*

Description

Performs Angular Randomisation Test for homogeneity on two samples of circular data after Ruxton et al. (2023). Test whether samples of directions share the mean direction P-values are estimated using permutation.

Usage

```
ar_test(x, y, n_perm = 1000L, axial = FALSE, alpha = NULL)
```

Arguments

`x, y`	numeric vectors. Angles in degrees
`n_perm`	integer. Number of permutations
`axial`	logical. Whether the data are axial, i.e. π-periodical (`TRUE`) or directional, i.e. 2π-periodical (`FALSE`, the default).
`alpha`	(optional) numeric. Significance level of the test (values between 0 and 1).

Details

Null Hypothesis (H_0): The two circular samples share an identical underlying probability distribution.

Alternative Hypothesis (H_1): The two samples come from different distributions.

Interpretation:

- Small p-value ($p < \alpha$, e.g., <0.05): Reject the null hypothesis. This indicates strong evidence that the two samples come from different circular distributions (differing in central tendency/mean direction or shape).
- Large p-value ($p \geq \alpha$): Fail to reject the null hypothesis; there is insufficient evidence to claim the two circular samples differ.

Value

A list with class `"htest"` containing the following components:

`statistic` the value of the G statistic

`parameter` the number of data in x and y, and the number of permutations

`p.value` the p-value for the test

`alternative` a character string describing the alternative hypothesis.

`method` a character string indicating what type of test was performed.

`alpha` the significance level

`reject` logical. whether to reject the null-hypothesis based on the significance level.

`data.name` a character string giving the name(s) of the data.

Note

Concentration Differences: The test can suffer from markedly lower statistical power if the underlying unimodal distributions differ by concentration (dispersion/spread) rather than location—especially with small, uneven sample sizes where the smaller sample comes from the more concentrated distribution.

Axial/Multimodal Data: ART performs poorly and loses power when applied to axially symmetric or symmetrically multimodal distributions.

References

Ruxton, G.D., Malkemper, E.P. & Landler, L. Evaluating the power of a recent method for comparing two circular distributions: an alternative to the Watson U2 test. Sci Rep 13, 10007 (2023). https://doi.org/10.1038/s41598-023-36960-1

See Also

Other Two-Sample-Tests: `equal_kappa_test()`, `uniform_scores_test()`, `walraff_test()`, `watson_two_sample`, `watson_wheeler_test_perm()`, `watson_williams_test()`

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
ar_test(x1, x2)

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
ar_test(sa.por$azi.PoR, rvm(100, 135, 10), axial = TRUE, alpha = 0.05)
```

`axes` *Plot axes*

Description

Show direction axes in a map

Usage

```
axes(
  x,
  y,
  angle,
  radius = 0.5,
  arrow.code = 1,
  arrow.length = 0,
  add = FALSE,
  ...
)
```

Arguments

x, y	coordinates of points
angle	Azimuth in degrees
radius	length of axis
arrow.code	integer. Kind of arrow head. The default is 1, i.e. no arrow head. See graphics::arrows() for details
arrow.length	numeric Length of the edges of the arrow head (in inches). (Ignored if arrow.code = 1)
add	logical. add to existing plot?
...	optional arguments passed to graphics::arrows()

Value

No return value, called for side effects

Examples

```
data("san_andreas")
axes(san_andreas$lon, san_andreas$lat, san_andreas$azi, add = FALSE)
```

---

circle_mean_diff *Circular Mean Difference*

Description

The circular mean difference is based on the sample circular distance

Usage

```
circular_mean_difference(x, w = NULL, axial = TRUE, na.rm = TRUE)

circular_mean_difference_alt(x, w = NULL, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric

References

Mardia, K.V., and Jupp, P.E (1999). Directional Statistics, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979

See Also

sample_circular_distance()

Examples

```
data("san_andreas")
circular_mean_difference(san_andreas$azi)
circular_mean_difference(san_andreas$azi, weighting(san_andreas$unc))

circular_mean_difference_alt(san_andreas$azi)
circular_mean_difference_alt(san_andreas$azi, weighting(san_andreas$unc))
```

---

circle_stats *Summary Statistics of Circular Data*

Description

Calculate the (weighted median) and standard deviation of orientation data.

Usage

```
circular_mean(x, w = NULL, axial = TRUE, na.rm = TRUE)

circular_var(x, w = NULL, axial = TRUE, na.rm = TRUE)

circular_sd(x, w = NULL, axial = TRUE, na.rm = TRUE)

circular_median(x, w = NULL, axial = TRUE, na.rm = TRUE)
```

```
circular_quantiles(x, w = NULL, axial = TRUE, na.rm = TRUE)

circular_IQR(x, w = NULL, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric vector

Note

Weighting may be the reciprocal of the data uncertainties.

Weightings have no effect on quasi-quantiles if `length(x) %% 4 == 0`.

References

Mardia, K.V. (1972). Statistics of Directional Data: Probability and Mathematical Statistics. London: Academic Press.

Mardia, K.V., and Jupp, P.E (1999). Directional Statistics, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979

N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.

Ziegler, M. O.; Heidbach O. (2019). Manual of the Matlab Script Stress2Grid v1.1. *WSM Technical Report* 19-02, GFZ German Research Centre for Geosciences. doi:10.2312/wsm.2019.002

Heidbach, O., Tingay, M., Barth, A., Reinecker, J., Kurfess, D., & Mueller, B. (2010). Global crustal stress pattern based on the World Stress Map database release 2008. *Tectonophysics* **482**, 3<U+2013>15, doi:10.1016/j.tecto.2009.07.023

Examples

```
set.seed(1)
x <- rvm(10, 0, 100) %% 180
unc <- stats::runif(length(x), 0, 10)
w <- weighting(unc)
circular_mean(x, w)
circular_var(x, w)
circular_sd(x, w)
circular_median(x, w)
circular_quantiles(x, w)
```

```
circular_IQR(x, w)

data("san_andreas")
w2 <- weighting(san_andreas$unc)
circular_mean(san_andreas$azi)
circular_mean(san_andreas$azi, w2)
circular_median(san_andreas$azi)
circular_median(san_andreas$azi, w2)
circular_quantiles(san_andreas$azi)
circular_quantiles(san_andreas$azi, w2)
circular_var(san_andreas$azi)
circular_var(san_andreas$azi, w2)

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_mean(sa.por$azi.PoR, w2)
circular_median(sa.por$azi.PoR, w2)
circular_var(sa.por$azi.PoR, w2)
circular_quantiles(sa.por$azi.PoR, w2)
```

`circular_density` *Circular Kernel Density Estimation*

Description

Kernel density estimates for circular data from a given kernel (von Mises, wrapped Cauchy, and wrapped Normal distribution and bandwidth

Usage

```
circular_density(
  x,
  z = NULL,
  bw = NULL,
  weights = NULL,
  na.rm = TRUE,
  from = 0,
  to = 360,
  n = 512L,
  axial = TRUE,
  kappa = NULL,
  rho = NULL,
  sd = NULL,
  c = NULL,
  kernel = c("vonmises", "wrappedcauchy", "wrappednormal", "wrappedlevy"),
  adjust = 1,
  subdensity = FALSE
)
```

Arguments

x	numeric. A vector of angles (in degrees) from which the estimate is to be computed.
z	numeric. Angles where the density is estimated. If NULL equally spaced angles are used according to the parameters from, to and n.
bw, kappa, rho, sd, c	numeric. Smoothing bandwidth expressed as the concentration parameter κ for the von Mises distribution, ρ for the wrapped Cauchy distribution, or σ for the wrapped normal distribution. Small and large values for the von Mises and wrapped normal/Cauchy distribution, respectively, gives smooth density lines. If not specified, parameter will be estimated using est.kappa() for the von Mises distribution, or set to $p\exp(-1)$ and 1 for the wrapped Cauchy and wrapped Normal distribution (where $p = 2$ when axial=TRUE and 1 otherwise), respectively.
weights	numeric. A vector of observation weights, of the same length as x, to give individual observations weight in the density estimate. Should sum to 1; a warning is issued if it doesn't (unless subdensity = TRUE). Defaults to equal weight 1/length(x) per observation, matching stats::density().
na.rm	logical; if TRUE, missing values are removed from x. If FALSE any missing values cause an error.
from, to	the left and right-most points of the grid at which the density is to be estimated; the defaults are cut * bw outside of range(x).
n	integer. Number of equally spaced angles at which the density is to be estimated.
axial	Logical. Whether data are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
kernel	character. The smoothing kernel to be used; one of "vonmises" (the default), "wrappedcauchy", "wrappednormal, for the von Mises, the wrapped Cauchy, and the wrapped Normal distribution.
adjust	the bandwidth used is actually adjust*bw. This makes it easy to specify values like 'half the default' bandwidth.
subdensity	logical. If TRUE, suppress the "sum(weights) != 1" warning, for when a deliberately partial (sub-)density is wanted, e.g. one group's contribution to a shared total. See stats::density().

Value

Object of class "density"

See Also

stats::density(), dvm(), dwcauchy(), dwnorm(), and plot_density()

Examples

```
w <-  weighting(san_andreas$unc)

# von Mises kernel density
```

```
circular_density(san_andreas$azi, kappa = 100)
circular_density(san_andreas$azi, weights = w, kappa = 100)

# wrapped Cauchy kernel density
circular_density(san_andreas$azi, rho = 0.9, kernel = "wrappedcauchy")

# wrapped Normal kernel density
circular_density(san_andreas$azi, sd = 5, kernel = "wrappednormal")
```

`circular_dispersion_boot` *Bootstrapped Estimates for Circular Dispersion*

Description

Calculates bootstrapped estimates of the circular dispersion, its standard error and its confidence interval.

Usage

```
circular_dispersion_boot(
  x,
  y = NULL,
  w = NULL,
  w.y = NULL,
  R = 1000,
  conf.level = 0.95,
  ...
)
```

Arguments

`x, y`	vectors of numeric values in degrees. `length(y)` is either `1` or `length(x)`
`w, w.y`	(optional) Weights. A vector of positive numbers and of the same length as `x`. `w.y` is the (optional) weight of `y`.
`R`	positive integer. The number of bootstrap replicates (1000 by default).
`conf.level`	Level of confidence: $(1 - \alpha\%)/100$. (`0.95` by default).
`...`	optional arguments passed to `boot::boot()`

Value

list containing:

`MLE` the maximum likelihood estimate of the circular dispersion

`sde` standard error of MLE

`CI` lower and upper limit of the confidence interval of MLE

See Also

circular_dispersion()

Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_dispersion(sa.por$azi.PoR, y = 135, w = weighting(san_andreas$unc))
circular_dispersion_boot(sa.por$azi.PoR, y = 135, w = weighting(san_andreas$unc), R = 1000)
```

---

circular_mode *Circular Mode*

Description

MLE angle (maximum density) using a circular distribution kernel with specified concentration

Usage

```
circular_mode(x, ...)
```

Arguments

x	numeric. A vector of angles (in degrees) from which the estimate is to be computed.
...	parameters passed to circular_density()

Value

numeric

References

N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.

Examples

```
set.seed(20250411)
x <- rvm(10, 0, 100)

# Mode of von Mises kernel density (the default)
circular_mode(x)

# Mode of wrapped Cauchy kernel density
circular_mode(x, kernel = "wrappedcauchy")
```

`circular_plot` *Circular plot*

Description

Polar diagram where angles increase clockwise.

Usage

```
circular_plot(
  main = NULL,
  sub = NULL,
  labels = TRUE,
  at = c(grid.lines, grid.lines + 180),
  cborder = TRUE,
  radius = 1,
  scale = 1.1,
  grid = FALSE,
  grid.lines = seq(0, 135, 45),
  grid.circles = seq(0.2, radius, 0.2),
  origin.text = "N",
  ...
)
```

Arguments

`main`	a main title for the plot, see also `title`.
`sub`	a subtitle for the plot.
`labels`	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
`at`	Optional vector of angles at which tick marks should be plotted. Set `at=numeric(0)` to suppress tick marks.
`cborder`	logical. Whether to plot the primitive of the circular plot (`TRUE` by default) or not.
`radius`	numeric. The radius of the circular plot. `1` by default.
`scale`	numeric. Increases the radius of the border to have some in-plot clearance. `1.1` by default
`grid`	logical. Whether to add a grid to the circular plot. Default is `FALSE`.
`grid.lines, grid.circles`	numeric. Adds a sequence of straight grid lines and circles based on angles (in degrees) and radii, respectively. Ignored when `grid=FALSE`
`origin.text`	character. String to be drawn at the top margin of the plot (`"N"` by default)
`...`	optional parameters passed to `graphics::symbols()` for plotting border of circular plot. Ignored if `cborder=FALSE`.

Value

none

Examples

```
circular_plot(main = "Title", grid = TRUE)
```

`circular_qqplot` *Quantile-Quantile Linearised Plot for Circular Distributions*

Description

Uniformly distributed orientations should yield a straight line through the origin. Systematic departures from linearity will indicate preferred orientation.

Usage

```
circular_qqplot(
  x,
  axial = TRUE,
  xlab = paste("i/(n+1)"),
  ylab = NULL,
  main = "Circular Quantile-Quantile Plot",
  add_line = TRUE,
  col = "#B63679FF",
  ...
)
```

Arguments

`x`	numeric. Angles in degrees
`axial`	Logical. Whether data are uniaxial (`axial=FALSE`)
`xlab`, `ylab`, `main`	plot labels.
`add_line`	logical. Whether to connect the points by straight lines?
`col`	color for the dots.
`...`	graphical parameters

Value

plot

References

Borradaile, G. J. (2003). Statistics of earth science data: their distribution in time, space, and orientation (Vol. 351, p. 329). Berlin: Springer.

See Also

cunif()

Other circ-qqplot: vm_qqplot()

Examples

```
set.seed(20250411)

# von Mises distribution
x_vm <- rvm(100, mean = 0, kappa = 2)
circular_qqplot(x_vm, pch = 20)

x_wcauchy <- rwcauchy(100, mean = 0, rho = 0.5)
circular_qqplot(x_wcauchy, pch = 20)

# circular uniform data
x_cunif <- rcunif(100)
circular_qqplot(x_cunif, pch = 20)
```

---

circular_range *Circular Range*

Description

Length of the smallest arc which contains all the observations. The circular range is based on the sample circular distance.

Usage

```
circular_range(x, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric. angle in degrees

References

Mardia, K.V., and Jupp, P.E (1999). Directional Statistics, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979

See Also

sample_circular_distance()

Examples

```
roulette <- c(43, 45, 52, 61, 75, 88, 88, 279, 357)
circular_range(roulette, axial = FALSE)

data("san_andreas")
circular_range(san_andreas$azi)
```

---

circular_sd_error *Standard Error of Mean Direction of Circular Data*

Description

Measure of the chance variation expected from sample to sample in estimates of the mean direction (after Mardia 1972). It is a parametric estimate of the the circular standard error of the mean direction by the particular form of the standard error for the von Mises distribution. The approximated standard error of the mean direction is computed by the mean resultant length and the MLE concentration parameter κ.

Usage

```
circular_sd_error(x, w = NULL, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric

References

- Batschelet, E. (1971). Recent statistical methods for orientation data. "Animal Orientation, Symposium 1970 on Wallops Island". Amer. Inst. Biol. Sciences, Washington.
- Mardia, K.V. (1972). Statistics of Directional Data: Probability and Mathematical Statistics. London: Academic Press.
- N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.
- Davis (1986) Statistics and data analysis in geology. 2nd ed., John Wiley & Sons.

See Also

mean_resultant_length(), circular_mean()

Examples

```
# Example data from Davis (1986), pp. 316
finland_stria <- c(
  23, 27, 53, 58, 64, 83, 85, 88, 93, 99, 100, 105, 113,
  113, 114, 117, 121, 123, 125, 126, 126, 126, 127, 127, 128, 128, 129, 132,
  132, 132, 134, 135, 137, 144, 145, 145, 146, 153, 155, 155, 155, 157, 163,
  165, 171, 172, 179, 181, 186, 190, 212
)
circular_sd_error(finland_stria, axial = FALSE)

data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_sd_error(sa.por$azi.PoR, w = weighting(san_andreas$unc))
```

---

circular_summary *Circular Summary Statistics*

Description

Circular mean, standard deviation, variance, quasi-quantiles, mode, 95% confidence angle, standardized skewness and kurtosis

Usage

```
circular_summary(
  x,
  w = NULL,
  axial = TRUE,
  mode = FALSE,
  kappa = NULL,
  fisher.CI = FALSE,
  conf.level = 0.95,
  na.rm = FALSE
)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).

mode	logical. Whether the circular mode should be calculated or not.
kappa	numeric. von Mises distribution concentration parameter used for the circular mode. Will be estimated using est.kappa() if not provided.
fisher.CI	logical. Whether Fisher's or the default Mardia/Batchelet's confidence interval should be calculated.
conf.level	Level of confidence: $(1 - \alpha\%)/100$. (0.95 by default).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

named vector

See Also

circular_mean(), circular_sd(), circular_var(), circular_quantiles(), confidence_angle(), second_central_moment(), circular_mode()

Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_summary(sa.por$azi.PoR)
circular_summary(sa.por$azi.PoR, w = weighting(san_andreas$unc))
```

---

compact-grid *Compact Smoothed Stress Field*

Description

Filter smoothed stress field containing a range of search radii or kernel half widths to find shortest wavelength (R) with the least circular sd. or dispersion (or any statistic) for each coordinate, respectively.

Usage

```
compact_grid(x, type = c("stress", "dispersion"))

compact_grid2(x, ..., FUN = min)
```

Arguments

x	output of stress2grid(), PoR_stress2grid(), stress2grid_stats(), or kernel_dispersion()
type	character. Type of the grid x. Either "stress" (when input is stress2grid() or PoR_stress2grid()) or "dispersion" (when input is kernel_dispersion()).
...	<tidy-select> One unquoted expression separated by commas. Variable names can be used as if they were positions in the data frame. Variable must be a column in x.
FUN	function is used to aggregate the data using the search radius R. Default is min().

Value

sf object

See Also

stress2grid(), PoR_stress2grid(), kernel_dispersion(), stress2grid_stats(), dplyr::dplyr_tidy_select()

Examples

```
data("san_andreas")
res <- stress2grid(san_andreas)
compact_grid(res) |> head()

## Not run:
res2 <- stress2grid_stats(san_andreas)
compact_grid2(res2, var, FUN = min)

## End(Not run)
```

---

confidence *Confidence Interval around the Mean Direction of Circular Data after Batschelet (1971)*

Description

Probabilistic limit on the location of the true or population mean direction, assuming that the estimation errors are normally distributed.

Usage

```
confidence_angle(x, conf.level = 0.95, w = NULL, axial = TRUE, na.rm = TRUE)

confidence_interval(x, conf.level = 0.95, w = NULL, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
conf.level	Level of confidence: $(1 - \alpha\%)/100$. (0.95 by default).
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Details

The confidence angle gives the interval, i.e. plus and minus the confidence angle, around the mean direction of a particular sample, that contains the true mean direction under a given level of confidence.

Value

Angle in degrees

References

- Batschelet, E. (1971). Recent statistical methods for orientation data. "Animal Orientation, Symposium 1970 on Wallops Island". Amer. Inst. Biol. Sciences, Washington.
- Mardia, K.V. (1972). Statistics of Directional Data: Probability and Mathematical Statistics. London: Academic Press. (p. 146)
- Davis (1986) Statistics and data analysis in geology. 2nd ed., John Wiley & Sons.
- Jammalamadaka, S. Rao and Sengupta, A. (2001). Topics in Circular Statistics, Sections 3.3.3 and 3.4.1, World Scientific Press, Singapore.

See Also

mean_resultant_length(), circular_sd_error()

Examples

```
# Example data from Davis (1986), pp. 316
finland_stria <- c(
  23, 27, 53, 58, 64, 83, 85, 88, 93, 99, 100, 105, 113,
  113, 114, 117, 121, 123, 125, 126, 126, 126, 127, 127, 128, 128, 129, 132,
  132, 132, 134, 135, 137, 144, 145, 145, 146, 153, 155, 155, 155, 157, 163,
  165, 171, 172, 179, 181, 186, 190, 212
)
confidence_angle(finland_stria, axial = FALSE)
confidence_interval(finland_stria, axial = FALSE)

data(san_andreas)
data("nuvel1")
```

```
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
confidence_angle(sa.por$azi.PoR, w = weighting(san_andreas$unc))
confidence_interval(sa.por$azi.PoR, w = weighting(san_andreas$unc))
```

`confidence_interval_fisher` *Confidence Interval around the Mean Direction of Circular Data after Fisher (1993)*

Description

For large samples (n >=25) i performs are parametric estimate based on `sample_circular_dispersion()`. For smaller size samples, it returns a bootstrap estimate.

Usage

```
confidence_interval_fisher(
  x,
  conf.level = 0.95,
  w = NULL,
  axial = TRUE,
  na.rm = TRUE,
  boot = FALSE,
  R = 1000L,
  quiet = FALSE
)
```

Arguments

`x`	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
`conf.level`	Level of confidence: $(1 - \alpha\%)/100$. (`0.95` by default).
`w`	(optional) Weights. A vector of positive numbers, of the same length as `x`.
`axial`	logical. Whether the data are axial, i.e. pi-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`).
`na.rm`	logical value indicating whether `NA` values in `x` should be stripped before the computation proceeds.
`boot`	logical. Force bootstrap estimation
`R`	positive integer. The number of bootstrap replicates (1000 by default).
`quiet`	logical. Prints the used estimation (parametric or bootstrap).

Value

list

References

N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.

Examples

```
# Example data from Davis (1986), pp. 316
finland_stria <- c(
  23, 27, 53, 58, 64, 83, 85, 88, 93, 99, 100, 105, 113,
  113, 114, 117, 121, 123, 125, 126, 126, 126, 127, 127, 128, 128, 129, 132,
  132, 132, 134, 135, 137, 144, 145, 145, 146, 153, 155, 155, 155, 157, 163,
  165, 171, 172, 179, 181, 186, 190, 212
)
confidence_interval_fisher(finland_stria, axial = FALSE)
confidence_interval_fisher(finland_stria, axial = FALSE, boot = TRUE)

data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
confidence_interval_fisher(sa.por$azi.PoR, w = weighting(san_andreas$unc))
confidence_interval_fisher(sa.por$azi.PoR, w = weighting(san_andreas$unc), boot = TRUE)
```

---

coordinates *Coordinate Transformations*

Description

Converts vector between Cartesian and geographical coordinate systems

Usage

```
cartesian_to_geographical(n)

geographical_to_cartesian(p)

geographical_to_spherical(p)
```

Arguments

n	Cartesian coordinates (x, y, z) as vector
p	Geographical coordinates (latitude, longitude) as vector

Value

Functions return a (2- or 3-dimensional) vector representing a point in the requested coordinate system.

See Also

cartesian_to_spherical() and spherical_to_cartesian() for conversions to spherical coordinates

Examples

```
n <- c(1, -2, 3)
cartesian_to_geographical(n)
p <- c(50, 10)
geographical_to_cartesian(p)
```

---

coordinates2 *Coordinate Transformations*

Description

Converts vector between Cartesian and spherical coordinate systems

Usage

```
cartesian_to_spherical(n)

spherical_to_cartesian(p)

spherical_to_geographical(p)
```

Arguments

n	Cartesian coordinates (x, y, z) as vector
p	Spherical coordinates (colatitude, azimuth) as two-column vector

Value

Functions return a (2- or 3-dimensional) vector representing a point in the requested coordinate system.

See Also

cartesian_to_geographical() and geographical_to_cartesian() for conversions to geographical coordinates

Examples

```
n <- c(1, -2, 3)
cartesian_to_spherical(n) # 36.699, -63.435
p <- c(50, 10)
spherical_to_cartesian(p) # 0.75, 0.13, 0.64
```

coordinate_mod *Coordinate Correction*

Description

Corrects the longitudes or latitudes to value between -180&deg and 180&deg or -90&deg and 90°

Usage

```
longitude_modulo(x)

latitude_modulo(x)
```

Arguments

x	Longitude(s) or latitude(s) in degrees

Value

numeric

Examples

```
longitude_modulo(-361 + 5 * 360) # -1
latitude_modulo(-91 + 5 * 180) # 89
```

cpm_models *Global model of current plate motions*

Description

Compilation of global models for current plate motions, including NUVEL1 (DeMets et al. 1990), NNR-NUVEL1A (DeMets et al., 1990), NNR-MORVEL56 (Argus et al., 2011), REVEL (Sella et al., 2002), GSRM2.1 (Kreemer et al., 2014), HS2-NUVEL1 (Gripp and Gordon, 1990), HS3-NUVEL1A (Gripp and Gordon, 2002), P073 (Chase 1978), AM1-2 (Minster and Jordan, 1978), ITRF2020-PPM (Altamimi et al. 2023), and PB2002 (Bird, 2003)

Usage

```
data('cpm_models')
```

Format

list containing objects of class `data.frame`

plate.name The rotating plate

plate.rot The abbreviation of the plate's name

lat,lon Coordinates of the Pole of Rotation

angle The amount of rotation (angle in 1 Myr)

plate.fix The anchored plate, i.e. `plate.rot` moves relative to `plate.fix`

model Model for current global plate motion

References

Altamimi, Z., Métivier, L., Rebischung, P., Collilieux, X., Chanard, K., Barnéoud, J., 2023. ITRF2020 Plate Motion Model. *Geophys. Res. Lett.* **50**, 1–7. doi:10.1029/2023GL106373

Argus, D.F., Gordon, R.G., 1991. No-net-rotation model of current plate velocities incorporating plate motion model NUVEL-1. *Geophys. Res. Lett.* **18**, 2039–2042. doi: 10.1029/91GL01532

Argus, D. F., Gordon, R. G., & DeMets, C. (2011). Geologically current motion of 56 plates relative to the no-net-rotation reference frame. *Geochemistry, Geophysics, Geosystems*, **12**(11). 10.1029/2011GC003751.

Chase, C.G. (1978). Plate kinematics: The Americas, East Africa, and the rest of the world. *Earth Planet. Sci. Lett.* **37**, 355–368. doi: doi:10.1016/0012821X(78)900511

Bird, P. (2003), An updated digital model of plate boundaries, *Geochem. Geophys. Geosyst.*, **4**, 1027, doi: 10.1029/2001GC000252, 3.

DeMets, C., Gordon, R. G., Argus, D. F., & Stein, S. (1990). Current plate motions. *Geophysical Journal International*, **101**(2), 425-478. doi:10.1111/j.1365246X.1990.tb06579.x.

Gripp, A. E., & Gordon, R. G. (2002). Young tracks of hotspots and current plate velocities. *Geophysical Journal International*, **150**(2), 321<U+2013>361. doi:10.1046/j.1365246X.2002.01627.x.

Kreemer, C., Blewitt, G., & Klein, E. C. (2014). A geodetic plate motion and Global Strain Rate Model. *Geochemistry, Geophysics, Geosystems*, **15**(10), 3849<U+2013>3889. doi: 10.1002/2014GC005407.

Minster, J. and Jorda, T. (1978). Present-day plate motions. *Journal of Geophysical Research*, **83**, doi:10.1029/jb083ib11p05331.

Sella, G. F., Dixon, T. H., & Mao, A. (2002). REVEL: A model for Recent plate velocities from space geodesy. *Journal of Geophysical Research: Solid Earth*, **107**(B4). doi: 10.1029/2000jb000033.

Examples

```
data("cpm_models")
head(cpm_models[[1]])
```

cunif *The Circular Uniform Distribution*

Description

Density, probability distribution function, quantiles, and random generation for the circular uniform distribution.

Usage

```
rcunif(n, axial = FALSE)

dcunif(theta, axial = FALSE, log = FALSE)

pcunif(theta, axial = FALSE, lower.tail = TRUE, log.p = FALSE)

qcunif(p, axial = FALSE, lower.tail = TRUE, log.p = FALSE)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
theta	numeric. Angular value in degrees
log, log.p	logical; if TRUE, probabilities/densities are given as logarithms.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
p	numeric. Vector of probabilities with values in $[0, 1]$.

Value

dcunif gives the density, pcunif gives the probability of circular uniform distribution function, rcunif generates random deviates (in degrees), and qcunif provides quantiles (in degrees).

See Also

wnorm, wcauchy, and vonmises

Examples

```
set.seed(1)
x <- rcunif(5)

dcunif(x)
dcunif(x, axial = TRUE)

pcunif(x)
qcunif(c(.25, .5, .75))
```

data2PoR *Transforms coordinates and azimuths into PoR coordinates system*

Description

Convenience function to add PoR coordinates and PoR azimuths to data

Usage

```
data2PoR(x, PoR)
```

Arguments

x	sf object or a data.frame containing the coordinates of the point(s) (lat, lon columns). x must contain the direction of σ_{Hmax} as column azi, its standard deviation (column unc) is optional).
PoR	data.frame or object of class euler.pole containing the geographical coordinates of the Euler pole.

Value

sf object in PoR CRS with additional columns lon.PoR, lat.PoR, and azi.PoR

Examples

```
por <- subset(nuvel1, nuvel1$plate.rot == "na")
data2PoR(san_andreas, por)
```

deviation_norm *Normalize Angle Between Two Directions*

Description

Normalizes the angle between two directions to the acute angle in between, i.e. angles between 0 and 90°

Usage

```
deviation_norm(x, y = NULL)
```

Arguments

x, y	Minuend and subtrahend. Both numeric vectors of angles in degrees. If y is missing, it treats x as difference. If not, length of subtrahend y is either 1 or equal to length(x).

Value

numeric vector, acute angles between two directions, i.e. values between 0 and 90°

Author(s)

Tobias Stephan

Examples

```
deviation_norm(175, 5)
deviation_norm(c(175, 95, 0), c(5, 85, NA))
deviation_norm(c(-5, 85, 95, 175, 185, 265, 275, 355, 365))
```

`deviation_shmax` *Deviation of Observed and Predicted Directions of Maximum Horizontal Stress*

Description

Calculate the angular difference between the observed and modeled direction of maximum horizontal stresses (σ_{Hmax}) along great circles, small circles, and loxodromes of the relative plate motion's Euler pole

Usage

```
deviation_shmax(prd, obs)
```

Arguments

`prd`	`data.frame` containing the modeled azimuths of σ_{Hmax}, i.e. the return object from `model_shmax()`
`obs`	Numeric vector containing the observed azimuth of σ_{Hmax}, same length as `prd`

Details

Deviation is positive for counterclockwise deviation of observed azimuth wrt. predicted azimuth.

Value

An object of class `data.frame`

dev.gc Deviation of observed stress from modeled σ_{Hmax} following great circles

dev.sc Small circles

dev.ld.cw Clockwise loxodromes

dev.ld.ccw Counter-clockwise loxodromes

Author(s)

Tobias Stephan

References

Stephan, T., Enkelmann, E., and Kroner, U. "Analyzing the horizontal orientation of the crustal stress adjacent to plate boundaries". *Sci Rep* 13. 15590 (2023). doi:10.1038/s41598023424332.

See Also

`model_shmax()` to calculate the theoretical direction of σ_{Hmax}.

Examples

```
data("nuvel1")
# North America relative to Pacific plate:
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")

# the point where we want to model the SHmax direction:
point <- data.frame(lat = 45, lon = 20)

prd <- model_shmax(point, PoR)
deviation_shmax(prd, obs = 90)
```

---

dispersion *Circular Distance and Dispersion*

Description

Circular distance between two angles and circular dispersion of angles about a specified angle.

Usage

```
circular_distance(x, y, axial = TRUE, na.rm = TRUE)

circular_dispersion(
  x,
  y = NULL,
  w = NULL,
  w.y = NULL,
  axial = TRUE,
  na.rm = TRUE
)

circular_sd2(x, y, w = NULL, axial = TRUE, na.rm = TRUE)
```

Arguments

x, y	vectors of numeric values in degrees. length(y) is either 1 or length(x)
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.
w, w.y	(optional) Weights. A vector of positive numbers and of the same length as x. w.y is the (optional) weight of y.

Details

Circular dispersion is a measure for the spread of data like the variance. Dispersion measures the spread about a given angles, whereas the variance measures the spread about the mean (Mardia and Jupp, 1999). When y = NULL the dispersion is identical to the variance.

Circular standard deviation in circular_sd2() is the transformed dispersion instead of the variance as for circular_sd().

Value

circular_distance returns a numeric vector of positive numbers, circular_dispersion and circular_sd2() return a positive number.

Note

If y is NULL, than the circular variance is returned.

References

Mardia, K.V. (1972). Statistics of Directional Data: Probability and Mathematical Statistics. London: Academic Press.

Mardia, K.V., and Jupp, P.E (1999). Directional Statistics, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979

See Also

circular_mean(), circular_var().

Examples

```
a <- c(0, 2, 359, 6, 354)
circular_distance(a, 10) # distance to single value

b <- a + 90
circular_distance(a, b) # distance to multiple values

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
```

```
circular_dispersion(sa.por$azi.PoR, y = 135)
circular_dispersion(sa.por$azi.PoR, y = 135, w = weighting(san_andreas$unc))
circular_sd2(sa.por$azi.PoR, y = 135, w = weighting(san_andreas$unc))
```

`distance_binned_stats` *Distance Binned Summary Statistics*

Description

[Experimental] Circular summary statistics over intervals of distances.

Usage

```
distance_binned_stats(
  azi,
  distance,
  n.breaks = 10,
  width.breaks = NULL,
  unc = NULL,
  prd = NULL,
  prd.error = NULL,
  kappa = 2,
  R = 1000,
  conf.level = 0.95,
  ...
)
```

Arguments

`azi`	numeric. Azimuth values in degrees.
`distance`	numeric. the independent variable along the values in `azi` are sorted, e.g. the plate boundary distances
`n.breaks`	numeric. number (greater than or equal to 2) giving the number of equal-sized intervals into which `distance` is to be cut. Default is 10. Will be ignored if `width.breaks` is specified.
`width.breaks`	numeric. The width of the intervals into which `distance` is to be cut.
`unc`	(optional) Uncertainties of `azi` (in degrees) acting as inverse weighting factors for statistics.
`prd`	(optional) numeric. A predicted orientation in degrees.
`prd.error`	(optional) numeric. The uncertainty of the predicted orientation in degrees.
`kappa`	numeric. Concentration parameter applied for the circular mode.
`R`	integer. Number of bootstrap iterates for estimating the error of the dispersion.
`conf.level`	The level of confidence for confidence interval and bootstrapped standard error of dispersion.
`...`	optional arguments passed to `ggplot2::cut_number()` and `ggplot2::cut_width()`

Value

tibble containing the n values for azi in each bin, min/median/max distance of the bin, and the summary statistics for azi. If prd is specified, the normal Chi-squared statistic, dispersion and its standard error are returned as well.

See Also

circular_summary(), circular_dispersion(), and circular_dispersion_boot()

Examples

```
data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
san_andreas$distance <- distance_from_pb(
  x = san_andreas,
  PoR = PoR,
  pb = plate_boundary,
  tangential = TRUE
)
dat <- san_andreas |> cbind(PoR_shmax(san_andreas, PoR, "right"))

distance_binned_stats(dat$azi.PoR,
  distance = dat$distance, width.breaks = 1,
  unc = dat$unc, prd = 135
) |> head()
```

---

distance_from_pb *Distance from plate boundary*

Description

Absolute distance of data points from the nearest plate boundary

Usage

```
distance_from_pb(x, PoR, pb, tangential = FALSE, km = FALSE, ...)
```

Arguments

x	sf or data.frame objects of the data points in geographical coordinate system
PoR	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Pole of Rotation
pb	sf objects of the plate boundary geometries in the geographical coordinate system
tangential	Logical. Whether the plate boundary is a tangential boundary (TRUE) or an inward and outward boundary (FALSE, the default).

km	Logical. Whether the distance is expressed in kilometers (TRUE) or in degrees (FALSE, the default).
...	optional arguments passed to smoothr::densify()

Details

The distance to the plate boundary is the longitudinal or latitudinal difference between the data point and the plate boundary (along the closest latitude or longitude) for inward/outward or tangential plate boundaries, respectively.

Value

Numeric vector of the great circle distances in units defined by km.

Note

Stresses emanate from the plate boundary along great circles, small circles or loxodromes associated with the pole of rotation. Hence the emanation distance is not necessarily the shortest distance to the plate boundary, which is measured along a great circle unrelated to the pole of rotation. The differences are particularly notable when the plate boundary is kinked or for convergent and divergent plate boundaries.

References

Wdowinski, S. (1998). A theory of intraplate tectonics. Journal of Geophysical Research: Solid Earth, 103(3), 5037<U+2013>5059. http://dx.doi.org/10.1029/97JB03390

Examples

```
data("nuvel1")
na_pa <- subset(nuvel1, nuvel1$plate.rot == "na")

data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")

data("san_andreas")
res <- distance_from_pb(
  x = san_andreas, PoR = na_pa, pb = plate_boundary, tangential = TRUE
)
head(res)

res.km <- distance_from_pb(
  x = san_andreas, PoR = na_pa, pb = plate_boundary, tangential = TRUE, km = TRUE
)
range(res.km)
```

dist_greatcircle *Distance between points*

Description

Returns the great circle distance between a location and all grid point in km

Usage

```
dist_greatcircle(
  lat1,
  lon1,
  lat2,
  lon2,
  r = earth_radius(),
  method = c("haversine", "orthodrome", "vincenty", "euclidean")
)
```

Arguments

lat1, lon1	numeric vector. coordinate of point(s) 1 (degrees).
lat2, lon2	numeric vector. coordinates of point(s) 2 (degrees).
r	numeric. radius of the sphere (default = 6371.0087714 km, i.e. the radius of the Earth)
method	Character. Formula for calculating great circle distance, one of:
	"haversine" great circle distance based on the haversine formula that is optimized for 64-bit floating-point numbers (the default)
	"orthodrome" great circle distance based on the spherical law of cosines
	"vincenty" distance based on the Vincenty formula for an ellipsoid with equal major and minor axes
	"euclidean" Euclidean distance (not great circle distance!)

Value

numeric vector with length equal to length(lat1) or length(lat2)

See Also

orthodrome(), haversine(), vincenty()

Examples

```
# Haversine: (4149.157, 2296.583) km
dist_greatcircle(lat1 = 20, lon1 = 12, lat2 = c(50, 30), lon2 = c(40, 32))

# Orthodrome: (4149.157, 2296.583) km
```

```
dist_greatcircle(
  lat1 = 20, lon1 = 12, lat2 = c(50, 30), lon2 = c(40, 32),
  method = "orthodrome"
)

# Vincenty: (4149.157, 2296.583) km
dist_greatcircle(
  lat1 = 20, lon1 = 12, lat2 = c(50, 30), lon2 = c(40, 32),
  method = "vincenty"
)

# Euclidean (4076.220, 2284.169) km
dist_greatcircle(
  lat1 = 20, lon1 = 12, lat2 = c(50, 30), lon2 = c(40, 32),
  method = "euclidean"
)
```

`earth_radius` *Earth's radius in km*

Description

IERS mean radius of Earth in km (based on WGS 84)

Usage

```
earth_radius()
```

Value

numeric value

`equal_kappa_test` *Equal κ Test*

Description

This function tests for the homogeneity of concentration parameters for multiple samples of directional data.

Usage

```
equal_kappa_test(samples, method = NULL, alpha = 0.05, axial = TRUE)
```

Arguments

`samples`	list of samples of orientations
`method`	character. One of
	`"f"` (default for two samples): $F = ((n_1 - R_1)/\nu_1)/((n_2 - R_2)/\nu_2)$, two-sided, $F(\nu_1, \nu_2)$ (Mardia and Jupp, 2000 eq. 10.5.3; Watson, 1956).
	`"bartlett"` (default for more samples): Bartlett's test of the dispersions $(n_i - R_i)/\nu_i$, chi-square with $q - 1$ degrees of freedom (Mardia and Jupp, 2000 eq. 10.6.21; McFadden & Lowes, 1981, eq. 64-66).
`alpha`	Significance level of the test ranging between 0 and 1. For `dist = "vonmises"` it must be one of `0.1`, `0.05`, `0.01` or `0` (only the tabulated p-value range is returned).
`axial`	logical. Whether the data are axial, i.e. π-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`). In case of axial data, the angles will be doubled for the test.

Details

The samples are assumed to have been drawn from von Mises populations. The null hypothesis tested is that all populations sampled have the same concentration parameter, kappa κ.

For concentrated data $2\kappa(n_i - R_i)$ is approximately chi-square with $\nu_i = n_i - 1$ degrees of freedom on the circle, where R_i is the resultant length of sample i.

Value

A list with class `"htest"`

Note

Both need concentrated data ($\bar{R} \geq 0.7$); MJ eq. 10.6.24-10.6.27 and 7.3.24-7.3.25 give variance-stabilised tests for dispersed data.

References

Jammalamadaka, S. Rao and SenGupta, A. (2001). Topics in Circular Statistics, Section 5.3, World Scientific Press, Singapore.

Mardia, K. and Jupp, P. (1999). Directional Statistics, Section 7.4, John Wiley and Sons, England.

See Also

Other Two-Sample-Tests: `ar_test()`, `uniform_scores_test()`, `walraff_test()`, `watson_two_sample`, `watson_wheeler_test_perm()`, `watson_williams_test()`

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
equal_kappa_test(list(x1, x2))

x1 <- rvm(100, mean = 90, kappa = 20)
x2 <- rvm(100, mean = 90, kappa = 10)
x3 <- rvm(100, mean = 90, kappa = 1)
equal_kappa_test(list(x1, x2, x3))

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
equal_kappa_test(list(sa.por$azi.PoR, rvm(100, 135, 10)))
```

---

equivalent_rotation *Equivalent rotation*

Description

Transforms a sequence of rotations into a new reference system

Usage

```
equivalent_rotation(x, fixed, rot)
```

Arguments

x	Object of class "data.frame" containing the Euler poles of plate rotations: plate.rot Moving plate lat, lon coordinates of Euler pole angle Angle of rotation plate.fix Fixed plate
fixed	plate that will be regarded as fixed. Has to be one out of x$plate.fix
rot	(optional) plate that will be regarded as rotating. Has to be one out of x$plate.rot.

Value

sequence of plate rotations in new reference system. Same object class as x

See Also

relative_rotation()

Examples

```
data(nuvel1) # load the NUVEL1 rotation parameters

# all nuvel1 rotation equivalent to fixed Africa:
equivalent_rotation(nuvel1, fixed = "af")
# relative plate motion between Eurasia and India:
equivalent_rotation(nuvel1, "eu", "in") # lat = 24.58, lon = 18.07, angle = 0.528
```

---

estimate-kappa *Concentration parameter of von Mises distribution*

Description

Estimates the concentration parameter of a von Mises distribution, given a set of angular measurements.

Usage

```
est.kappa.MLE(x, w = NULL, bias = FALSE)

est.kappa(x, w = NULL, p = 2)
```

Arguments

x	numeric. Angles in degrees
w	numeric. Weightings
bias	logical parameter determining whether a bias correction is used in the computation of the MLE. Default for bias is FALSE for no bias correction.
p	integer. Number of parameters in the data space: 2 for circle (the default), 3 for a sphere.

Details

`est.kappa.MLE()` is the maximum likelihood estimate for MLE for κ.

`est.kappa()` uses an approximation based on the empirical equation:

$$\kappa = \frac{\bar{R}(p - \bar{R}^2)}{1 - \bar{R}^2}$$

where $\bar{R}$ is the mean resultant length and p is the dimensionality of the data (2 for circular data).

Value

numeric. Concentration of a von Mises distribution

See Also

vonmises()

Examples

```
set.seed(123)
x <- rvm(100, 90, 10)
w <- weighting(runif(100, 0, 10))

est.kappa(x, w)

est.kappa.MLE(x, w)
```

`euler_pole` *Euler pole object*

Description

Creates an object of the orientation of the Euler pole axis

Usage

```
euler_pole(x, y, z = NA, geo = TRUE, angle = NA)
```

Arguments

`x`	latitude or x coordinate of Euler pole axis
`y`	longitude or y
`z`	z coordinate
`geo`	logical, `TRUE` (the default) if Euler pole axis is given in geographical coordinates (latitude, longitude). `FALSE` if given in Cartesian coordinates (`x`, `y`, `z`)
`angle`	(optional) Angle of rotation in degrees (CCW rotation if angle is positive)

Value

An object of class `"euler.pole"` containing the Euler pole axis in both geographical and Cartesian coordinates and the angle of rotation in radians.

Examples

```
euler_pole(90, 0, angle = 45)
euler_pole(0, 0, 1, geo = FALSE)
```

geom_azimuth *Azimuth visualization*

Description

geom_azimuth() visualizes axial-directional vector fields using a geom to produce a new graphical layer, which allows aesthetic options. This layer can be overlaid on a map to improve visualisation of mapped data. The geom draws line segments (spokes) centered at (x, y) with a given orientation (angle in degrees) and length (radius). By default the spoke is centered using PositionCenterSpoke, so that the given coordinates mark the middle of the line. The azimuths are given as angles in degrees increasing clockwise from North.

Usage

```
geom_azimuth(
  mapping = NULL,
  data = NULL,
  stat = "azimuth",
  center = TRUE,
  radius = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes().
data	A data frame. If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot2::ggplot().
stat	The statistical transformation to use on the data. Defaults to "identity".
center	Logical; if TRUE (the default) spokes are centered on (x, y) using PositionCenterSpoke - useful for axial data. If FALSE, behaves like ggplot2::geom_spoke() (line starts at (x, y)) - useful for directional data (especially when in combination with arrow()).
radius	Length of spoke
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them.
...	Other arguments passed on to ggplot2::layer(). These are often aesthetics (e.g. colour, linetype, linewidth, alpha).

Value

A ggplot2 layer that adds axis-like spokes.

Aesthetics

`geom_azimuth()` understands the following aesthetics (required aesthetics in **bold**):

- **x**
- **y**
- angle (in degrees, transformed internally)
- radius
- colour
- alpha
- linewidth
- linetype

See Also

`ggplot2::geom_spoke()`, `geom_azimuthpoint()`

Examples

```
set.seed(20250411)
df <- data.frame(
  x = runif(5), y = runif(5),
  angle_deg = rvm(5, mean = 90, kappa = 10),
  radius = runif(5, 0.1, 2)
)

if (require("ggplot2")) {
  ggplot(df, aes(x, y)) +
    geom_azimuth(aes(angle = angle_deg), radius = .1, linewidth = 1.2, colour = "blue")
  if (require("grid")) {
    ggplot(df, aes(x, y, radius = radius)) +
    geom_azimuth(aes(angle = angle_deg), center = FALSE, colour = "red", arrow = grid::arrow())
  }
}
```

`geom_azimuthpoint` *Azimuth + point visualization*

Description

`geom_azimuthpoint()` draws line segments (spokes) like `geom_azimuth()`, but also places a point (marker) at the spoke's center `(x, y)`.

Aesthetic rules:

- `linewidth`, `linetype` affect the spoke only
- `shape` affects the point only
- `colour`, `alpha` affect both spoke and point
- `size` sets the size of the point only

Usage

```
geom_azimuthpoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  center = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  size = 2,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `ggplot2::aes()`.
`data`	A data frame. If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot2::ggplot()`.
`stat`	The statistical transformation to use on the data. Defaults to `"identity"`.
`center`	Logical; if `TRUE` spokes are centered on (x, y) using `PositionCenterSpoke`. If `FALSE`, behaves like `ggplot2::geom_spoke()` (line starts at (x, y)).
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	Logical. Should this layer be included in the legends?
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them.
`size`	Size of the point marker (default = 2).
`...`	Other arguments passed on to `geom_azimuth()` and `ggplot2::geom_point()`. These may include `arrow`, `fill`, etc.

Value

A list of ggplot2 layers (spokes + points).

Aesthetics

`geom_azimuthpoint()` understands the following aesthetics (required aesthetics in **bold**):

- **x**
- **y**
- angle (in degrees, transformed internally; spoke only)
- radius (spoke only)
- colour (shared)
- alpha (shared)
- linewidth (spoke only)
- linetype (spoke only)
- shape (point only)
- size (point only, or via argument)
- fill (point only, for shapes that accept fill)

See Also

`geom_azimuth()`, `ggplot2::geom_spoke()`, `ggplot2::geom_point()`

Examples

```
set.seed(20250411)
df <- data.frame(
  x = runif(5), y = runif(5),
  angle_deg = rvm(5, mean = 90, kappa = 10),
  radius = runif(5, 0.1, 1),
  group = rep(1:2, length.out = 5)
)

if (require("ggplot2")) {
  ggplot(data = df, aes(x, y, angle = angle_deg, radius = radius)) +
    geom_azimuthpoint(aes(colour = factor(group), shape = factor(group)),
      linewidth = 1.1, linetype = "dashed",
      size = 3, alpha = 0.8
    )
}
```

`get_azimuth` *Azimuth Between two Points*

Description

Calculate initial bearing (or forward azimuth/direction) to go from point `a` to point `b` following great circle arc on a sphere.

Usage

```
get_azimuth(lat_a, lon_a, lat_b, lon_b)
```

Arguments

lat_a, lat_b	Numeric. Latitudes of a and b (in degrees).
lon_a, lon_b	Numeric. Longitudes of a and b (in degrees).

Details

get_azimuth() is based on the spherical law of tangents. This formula is for the initial bearing (sometimes referred to as forward azimuth) which if followed in a straight line along a great circle arc will lead from the start point a to the end point b.

$$\theta = \arctan 2(\sin \Delta\lambda \cos \psi_2, \cos \psi_1 \sin \psi_1 - \sin \psi_1 \cos \psi_2 \cos \Delta\lambda)$$

where ψ_1, λ_1 is the start point, ψ_2, λ_2 the end point ($\Delta\lambda$ is the difference in longitude).

Value

numeric. Azimuth in degrees

References

http://www.movable-type.co.uk/scripts/latlong.html

Examples

```
berlin <- c(52.517, 13.4) # Berlin
tokyo <- c(35.7, 139.767) # Tokyo
get_azimuth(berlin[1], berlin[2], tokyo[1], tokyo[2]) # 41.57361
```

---

hodges_ajne_test *Hodges-Ajne Test*

Description

Hodges-Ajne test of uniformity on the circle

Usage

```
hodges_ajne_test(x, alpha = 0.05, axial = TRUE)
```

Arguments

x	numeric vector. Values in degrees
alpha	Significance level of the test ranging between 0 and 1.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.

Details

The statistic m is the smallest number of angles in a half-circle. Its exact distribution is

$$P(m \leq t) = 2^{(1-n)}(n - 2t)C(n, t)$$

for $t < n/3$ (Mardia and Jupp, 2000, sec. 6.3.4), which gives the p-value (1 above $n/3$). Small m indicates a preferred direction.

Value

A list with class `"htest"` containing the following components:

`statistic` the value of the m statistic

`parameter` the number of data

`p.value` the p-value for the test

`alternative` a character string describing the alternative hypothesis.

`method` a character string indicating what type of test was performed.

`data.name` a character string giving the name(s) of the data.

See Also

Other One-Sample-Tests: `kuiper_test()`, `norm_chisq()`, `rao_spacing_test()`, `rayleigh-test`, `watson_test()`, `weighted-rayleigh-test`

Examples

```
hodges_ajne_test(rvm(100, mean = 0, kappa = 10))

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
hodges_ajne_test(sa.por$azi.PoR, alpha = .05)
```

---

homing *Homing directions of Pidgeons*

Description

Directions of 10 pidgeons, example data from Mardia and Jupp (1999), pp. 93

Format

numeric vector of length 10

References

Mardia and Jupp (1999). Directional Statistics. John Wiley and Sons.

Examples

```
data("homing")
homing
```

`import_WSM` *World Stress Map Database (WSM)*

Description

Download WSM2025 or WSM2016 database from the GFZ sever and applies optional filters. If `destdir` is specified, the data can be reloaded in a later R session using `load_WSM()` using the same arguments.

Usage

```
download_WSM(
  destdir = tempdir(),
  load = TRUE,
  version = c("2025", "2016"),
  ...
)

load_WSM(
  file,
  quality = c("A", "B", "C", "D", "E", "X"),
  lat_range = c(-90, 90),
  lon_range = c(-180, 180),
  depth_range = c(-Inf, Inf),
 type = c("BO", "BOC", "BOT", "BS", "DIF", "FMA", "FMF", "FMS", "GFI", "GFM", "GFS",
  "GVA", "HF", "HFG", "HFM", "HFH", "HFP", "HFS", "OC", "PC", "SWB", "SWL", "SWS"),
  regime = c("N", "NS", "T", "TS", "S", NA)
)

download_WSM2016(destdir = tempdir(), load = TRUE, ...)

load_WSM2016(
  file,
  quality = c("A", "B", "C", "D", "E"),
  lat_range = c(-90, 90),
  lon_range = c(-180, 180),
  depth_range = c(-Inf, Inf),
 type = c("BO", "BOC", "BOT", "BS", "DIF", "FMA", "FMF", "FMS", "GFI", "GFM", "GFS",
    "GVA", "HF", "HFG", "HFM", "HFP", "OC", "PC", "SWB", "SWL", "SWS"),
```

```
  regime = c("N", "NS", "T", "TS", "S", NA)
)
```

Arguments

destdir	where to save files, defaults to base::tempdir(), base::getwd() is also possible.
load	TRUE load the dataset into R, FALSE return the file name of the downloaded object.
version	character. Version of the World stress map database. Either "2025" (default) or "2016"
...	(optional) arguments passed to load_WSM()
file	the name of the file which the data are to be read from.
quality	a character vectors containing the quality levels to be included. Includes all quality ranks (A-X) by default.
lat_range, lon_range	two-element numeric vectors giving the range of latitudes and longitudes (in degrees).
depth_range	two-element numeric vectors giving the depth interval (in km)
type	a character vectors containing the methods of stress inversion to be included. Includes all methods by default. See WSM2016 manual for used acronyms.
regime	a character vectors containing the stress regimes to be included. Acronyms: "N" - normal, "T" - thrust, "S" - strike-slip, "NS" - oblique normal, "TS" - oblique thrust, and NA - unknown faulting

Value

sf object of and the parsed numeric uncertainty (unc) based on the reported standard deviation and the quality rank. If load=FALSE, the path to the downloaded file is returned.

Note

Because of R-compatibility and easy readability reasons, the downloaded dataset is a modified version of the original, WSM server version: All column names have been changed from uppercase (in the original dataset) to lowercase characters. Unknown azimuth values are represented by NA values instead of 999 in the original. Unknown regimes are represented by NA instead of "U" in the original.

Source

https://datapub.gfz.de/download/10.5880.WSM.2025.001-Scbwez/WSM_Database_2025.csv

https://datapub.gfz-potsdam.de/download/10.5880.WSM.2016.001/wsm2016.csv

References

Heidbach, O., M. Rajabi, X. Cui, K. Fuchs, B. M<U+00FC>ller, J. Reinecker, K. Reiter, M. Tingay, F. Wenzel, F. Xie, M. O. Ziegler, M.-L. Zoback, and M. D. Zoback (2018): The World Stress Map database release 2016: Crustal stress pattern across scales. *Tectonophysics*, **744**, 484-498, doi:10.1016/j.tecto.2018.07.007.

Heidbach, Oliver; Rajabi, Mojtaba; Di Giacomo, Domenico; Harris, James; Lammers, Steffi; Morawietz, Sophia; Pierdominici, Simona; Reiter, Karsten; von Specht, Sebastian; Storchak, Dmitry; Ziegler, Moritz O. (2025): World Stress Map Database Release 2025. GFZ Data Services. doi:10.5880/WSM.2025.001

Examples

```
## Not run:
download_WSM(
  quality = c("A", "B", "C"), lat_range = c(51, 72),
  lon_range = c(-180, -130), depth_range = c(0, 10), type = "FMS"
)

## End(Not run)
```

`is.euler` *Check if object is euler.pole*

Description

Check if object is euler.pole

Usage

```
is.euler(x)
```

Arguments

`x`	Object of class `"data.frame"` containing the Euler poles of plate rotations: `plate.rot` Moving plate `lat`, `lon` coordinates of Euler pole `angle` Angle of rotation `plate.fix` Fixed plate

Value

logical

`kuiper_test` *Kuiper Test of Circular Uniformity*

Description

A statistical test used to determine whether a set of angular or circular data points (such as times of day, compass directions, or degrees) are spread out evenly around a circle or if they cluster in some way.

Usage

```
kuiper_test(x, alpha = 0.05, axial = TRUE, quiet = FALSE)
```

Arguments

`x`	numeric vector. Values in degrees
`alpha`	Significance level of the test ranging between 0 and 1.
`axial`	logical. Whether the data are axial, i.e. π-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`). In case of axial data, the angles will be doubled for the test.
`quiet`	logical. Prints the test's decision.

Details

The **Null Hypothesis** (H_0): The data are distributed completely uniformly (randomly and evenly) around the circle.

The **Alternative Hypothesis** (H_1): The data are not uniform and show a preference, clustering, or pattern somewhere on the circle.

The Test Statistic (V or $D^+ + D^-$): It measures the greatest positive and negative differences between your data's empirical cumulative distribution and a theoretical uniform distribution.

Interpreting the Results:

- High Test Statistic / Low p-value ($p < \alpha$, typically 0.05): You reject the null hypothesis. This means your data are not uniform; they have a significant preferred direction, grouping, or non-random pattern on the circle.
- Low Test Statistic / High p-value ($p \geq 0.05$): You fail to reject the null hypothesis. There is no strong evidence to say the data are different from a flat, uniform distribution. The points appear random across the circle.

Value

A list with class `"htest"` containing the following components:

`statistic` the value of the modified V statistic

`parameter` the number of data

`p.value` the p-value for the test

`estimate` the D+ and D- values

`alternative` a character string describing the alternative hypothesis.

`method` a character string indicating what type of Kuiper test was performed.

`data.name` a character string giving the name(s) of the data.

`critical` the critical value at the significance level

`alpha` the significance level

`reject` logical. whether to reject the null-hypothesis based on the significance level.

Note

Kuiper's test statistic is a rotation-invariant Kolmogorov-type test statistic. The modified statistic of Stephens (1970) is compared with the asymptotic Kuiper distribution.

See Also

Other One-Sample-Tests: `hodges_ajne_test()`, `norm_chisq()`, `rao_spacing_test()`, `rayleigh-test`, `watson_test()`, `weighted-rayleigh-test`

Examples

```
# Example data from Mardia and Jupp (1999), pp. 93
kuiper_test(homing, alpha = .05)

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
kuiper_test(sa.por$azi.PoR, alpha = .05)
```

`line_azimuth` *Extract azimuths of line segments*

Description

Extract azimuths of line segments

Usage

```
line_azimuth(x, warn = TRUE)

lines_azimuths(x)
```

Arguments

x	sf object of type "LINESTRING" or "MULTILINESTRING"
warn	logical; if TRUE, warn if "MULTILINESTRING" (default).

Details

It is recommended to perform line_azimuth() on single line objects, i.e. type "LINESTRING", instead of "MULTILINESTRING". This is because the azimuth of the last point of a line will be calculated to the first point of the next line otherwise. This will cause a warning message (if warn = TRUE). For "MULTILINESTRING" objects, use lines_azimuths().

Value

sf object of type "POINT" with the columns and entries of the first row of x

Examples

```
data("plates")

# one line:
subset(plates, pair == "af-eu") |>
  smoothr::densify() |>
  line_azimuth() |>
  head()

# multiple lines:
lines_azimuths(plates[1:5, ]) |> head()
```

---

mean_resultant_length *Mean Resultant Length*

Description

Measure of spread around the circle. It should be noted that: If $R = 0$, then the data is completely spread around the circle. If $R = 1$, the data is completely concentrated on one point.

Usage

```
mean_resultant_length(x, w = NULL, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric.

References

Mardia, K.V. (1972). Statistics of Directional Data: Probability and Mathematical Statistics. London: Academic Press.

Examples

```
# Example data from Davis (1986), pp. 316
finland_stria <- c(
  23, 27, 53, 58, 64, 83, 85, 88, 93, 99, 100, 105, 113,
  113, 114, 117, 121, 123, 125, 126, 126, 126, 127, 127, 128, 128, 129, 132,
  132, 132, 134, 135, 137, 144, 145, 145, 146, 153, 155, 155, 155, 157, 163,
  165, 171, 172, 179, 181, 186, 190, 212
)
mean_resultant_length(finland_stria, w = NULL, na.rm = FALSE) # 0.800
```

`model_shmax` *Theoretical Direction of Maximum Horizontal Stress in the geographical reference system.*

Description

Models the direction of maximum horizontal stress σ_{Hmax} along great circles, small circles, and loxodromes at a given point or points according to the relative plate motion in the geographical coordinate reference system.

Usage

```
model_shmax(df, euler)
```

Arguments

`df`	`data.frame` containing the coordinates of the point(s) (`lat`, `lon`).
`euler`	`"data.frame"` or object of class `"euler.pole"` containing the geographical coordinates of the Euler pole

Details

σ_{Hmax} following *great circles* is the (initial) bearing between the given point and the pole of relative plate motion. σ_{Hmax} along *small circles*, clockwise, and counter-clockwise *loxodromes* is 90°, +45°, and +135° (or -45°) to this great circle bearing, respectively.

Value

`data.frame`

gc Azimuth of modeled σ_{Hmax} following great circles

sc Small circles

ld.cw Clockwise loxodromes

ld.ccw Counter-clockwise loxodromes

Author(s)

Tobias Stephan

References

Stephan, T., Enkelmann, E., and Kroner, U. "Analyzing the horizontal orientation of the crustal stress adjacent to plate boundaries". *Sci Rep* 13. 15590 (2023). doi:10.1038/s41598023424332.

See Also

`deviation_shmax()` to compute the deviation of the modeled direction from the observed direction of σ_{Hmax}. `PoR_shmax()` to calculate the azimuth of σ_{Hmax} in the pole of rotation reference system.

Examples

```
data("nuvel1")
# North America relative to Pacific plate:
euler <- subset(nuvel1, nuvel1$plate.rot == "na")

# the point where we mant to model the SHmax direction:
point <- data.frame(lat = 45, lon = 20)

model_shmax(point, euler)
```

---

norm_chisq *Normalized Chi-Squared Test for Circular Data*

Description

A quantitative comparison between the predicted and observed directions of σ_{Hmax} is obtained by the calculation of the average azimuth and by a normalized χ^2 test (Wdowsinki, 1998)

Usage

```
norm_chisq(obs, prd, unc)
```

Arguments

obs	Numeric vector containing the observed azimuth of σ_{Hmax}, same length as prd
prd	Numeric vector containing the modeled azimuths of σ_{Hmax}, i.e. the return object from model_shmax()
unc	Uncertainty of observed σ_{Hmax}, either a numeric vector or a number. Observations with missing uncertainty are dropped; an uncertainty of 0 is replaced by 1.

Details

The normalized χ^2 test is

$$\text{Norm}\chi_i^2 = \frac{\sum_{i=1}^{M}\left(\frac{\alpha_i - \alpha_{predict}}{\sigma_i}\right)^2}{\sum_{i=1}^{M}\left(\frac{90}{\sigma_i}\right)^2}$$

The value of the chi-squared test statistic is a number between 0 and 1 indicating the quality of the predicted σ_{Hmax} directions. Low values (≤ 0.15) indicate good agreement, high values (> 0.7) indicate a systematic misfit between predicted and observed σ_{Hmax} directions.

Value

Numeric vector

References

Wdowinski, S., 1998, A theory of intraplate tectonics. *Journal of Geophysical Research: Solid Earth*, **103**, 5037-5059, doi: 10.1029/97JB03390.

See Also

Other One-Sample-Tests: hodges_ajne_test(), kuiper_test(), rao_spacing_test(), rayleigh-test, watson_test(), weighted-rayleigh-test

Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to
# Pacific plate
data(san_andreas)
point <- data.frame(lat = 45, lon = 20)
prd <- model_shmax(point, PoR)
norm_chisq(obs = c(50, 40, 42), prd = prd$sc, unc = c(10, NA, 5))

data(san_andreas)
prd2 <- PoR_shmax(san_andreas, PoR, type = "right")
norm_chisq(obs = prd2$azi.PoR, 135, unc = san_andreas$unc)
```

nuvel1 *NUVEL-1 Global model of current plate motions*

Description

NNR-NUVEL-1 global model of current plate motions by DeMets et al. 1990

Usage

```
data('nuvel1')
```

Format

An object of class `data.frame`

plate.name The rotating plate

plate.rot The abbreviation of the plate's name

lat,lon Coordinates of the Pole of Rotation

angle The amount of rotation (angle in 1 Myr)

plate.fix The anchored plate, i.e. `plate.rot` moves relative to `plate.fix`

source Reference to underlying study

References

DeMets, C., Gordon, R. G., Argus, D. F., & Stein, S. (1990). Current plate motions. *Geophysical Journal International*, **101**(2), 425-478. doi:10.1111/j.1365246X.1990.tb06579.x.

Examples

```
data("nuvel1")
head("nuvel1")
```

nuvel1_plates *Plate Boundaries on the Earth*

Description

Global set of present plate boundaries on the Earth based on NUVEL-1 model by DeMets et al. 1990

Usage

```
data('nuvel1_plates')
```

Format

An object of class `sf`

References

DeMets, C., Gordon, R. G., Argus, D. F., & Stein, S. (1990). Current plate motions. *Geophysical Journal International*, **101**(2), 425-478. doi:10.1111/j.1365246X.1990.tb06579.x.

Examples

```
data("nuvel1_plates")
head("nuvel1_plates")
```

---

ort-eigen *Decomposition of Orientation Tensor in 2D*

Description

Spectral decomposition of the 2D orientation tensor into two Eigenvectors and corresponding Eigenvalues provides provides a measure of location and a corresponding measure of dispersion, respectively.

Usage

```
ot_eigen2d(x, w = NULL, scale = FALSE)

principal_direction(x, w = NULL)

axial_strength(x, w = NULL)

axial_dispersion(x, w = NULL)
```

Arguments

`x`	numeric. Axial angular data (in degrees).
`w`	(optional) Weights. A vector of positive numbers, of the same length as `x`.
`scale`	logical. Whether the Eigenvalues should be scaled so they sum up to 1. Only applicable when weighting are specified, otherwise the eigenvalues are always scaled.

Details

The **Eigenvalues** ($\lambda_1 > \lambda_2$) can be interpreted as the fractions of the variance explained by the orientation of the associated Eigenvectors. The two perpendicular **Eigenvectors** (a_1, a_2) are the "principal directions" with respect to the highest and the lowest concentration of orientation data.

The strength of the orientation is the largest eigenvalue λ_1 normalized by the sum of the eigenvalues (`scale=TRUE`). Then $\lambda_2 = 1 - \lambda_1$ is a **measure of dispersion** of 2D orientation data with respect to a_1.

Value

ot_eigen2d returns a list of the Eigenvalues and the axial angles corresponding to the Eigenvectors. principal_direction(), axial_strength() and axial_dispersion() are convenience functions to return the orientation of the largest eigenvalue, the orientation strength, the axial dispersion respectively.

Note

Eigenvalues and Eigenvectors of the orientation tensor (inertia tensor) are also called "principle moments of inertia" and "principle axes of inertia", respectively.

See Also

ortensor2d()

Examples

```
test <- rvm(100, mean = 0, k = 10) / 2
ot_eigen2d(test)

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
sa_eig <- ot_eigen2d(sa.por$azi.PoR, w = weighting(san_andreas$unc), scale = TRUE)
print(sa_eig)

rose(sa.por$azi.PoR, muci = FALSE)
rose_line(sa_eig$vectors,
  col = c("red", "green"),
  radius = sa_eig$values, lwd = 2
)
graphics::legend("topright",
  legend = round(sa_eig$values, 2),
  col = c("red", "green"), lty = 1
)

principal_direction(sa.por$azi.PoR)

axial_strength(sa.por$azi.PoR)

axial_dispersion(sa.por$azi.PoR)
```

---

ortensor2d *Orientation Tensor*

Description

2D orientation tensor characterizes distribution of axial angles using the Eigenvalue method (Watson 1966, Scheidegger 1965).

Usage

```
ortensor2d(x, w = NULL, norm = FALSE)
```

Arguments

x	numeric. Axial angular data (in degrees).
w	(optional) Weights. A vector of positive numbers, of the same length as x.
norm	logical. Whether the tensor should be normalized.

Details

The moment of inertia can be minimized by calculating the Cartesian coordinates of the orientation data, and calculating their covariance matrix. This yields

$$I = x \cdot x^{\intercal}$$

where x is the Cartesian vector of the orientations. Orientation tensor T and the inertia tensor I are related by

$$I = E - T$$

where E denotes the unit matrix, so that

$$T = \frac{1}{n} \sum_{i=i}^{n} x_i \cdot x_i^{\intercal}$$

Value

2x2 matrix

References

Watson, G. S. (1966). The Statistics of Orientation Data. The Journal of Geology, 74(5), 786–797.

Scheidegger, A. E. (1964). The tectonic stress and tectonic motion direction in Europe and Western Asia as calculated from earthquake fault plane solutions. Bulletin of the Seismological Society of America, 54(5A), 1519–1528. doi:10.1785/BSSA05405A1519

Bachmann, F., Hielscher, R., Jupp, P. E., Pantleon, W., Schaeben, H., & Wegert, E. (2010). Inferential statistics of electron backscatter diffraction data from within individual crystalline grains. Journal of Applied Crystallography, 43(6), 1338–1355. https://doi.org/10.1107/S002188981003027X

See Also

ot_eigen2d()

Examples

```
test <- rvm(100, mean = 0, k = 10)
ortensor2d(test)
```

parse_wsm *Numerical values to World Stress Map Quality Ranking*

Description

Assigns numeric values of the precision (sd.) of each measurement to the categorical quality ranking of the World Stress Map (A, B, C, D, E, X).

Usage

```
parse_wsm_quality(x)

quantise_wsm_quality(x)
```

Arguments

x	Either a string or a character/factor vector of WSM quality ranking

Value

"numeric". the standard deviation of stress azimuth

References

Heidbach, O., Barth, A., M<U+00FC>ller, B., Reinecker, J., Stephansson, O., Tingay, M., Zang, A. (2016). WSM quality ranking scheme, database description and analysis guidelines for stress indicator. *World Stress Map Technical Report* **16-01**, GFZ German Research Centre for Geosciences. doi:10.2312/wsm.2016.001

Examples

```
parse_wsm_quality(c("A", "B", "C", "D", NA, "E", "X"))
data("san_andreas")
head(parse_wsm_quality(san_andreas$quality))
```

pb2002 *Global model of current plate motions*

Description

PB2002 global model of current plate motions by Bird 2003

Usage

```
data('pb2002')
```

Format

An object of class `data.frame`

plate.name The rotating plate

plate.rot The abbreviation of the plate's name

lat,lon Coordinates of the Pole of Rotation

angle The amount of rotation (angle in 1 Myr)

plate.fix The anchored plate, i.e. `plate.rot` moves relative to `plate.fix`

source Reference to underlying study

References

Bird, P. (2003), An updated digital model of plate boundaries, *Geochem. Geophys. Geosyst.*, **4**, 1027, doi: 10.1029/2001GC000252, 3.

Examples

```
data("pb2002")
head("pb2002")
```

`plates` *Plate Boundaries on the Earth*

Description

Global set of present plate boundaries on the Earth based on PB2002 model by Bird (2003). Contains the plate boundary displacement types such as inward, outward, or tangentially displacement.

Usage

```
data('plates')
```

Format

An object of class `sf`

References

Bird, P. (2003), An updated digital model of plate boundaries, *Geochem. Geophys. Geosyst.*, **4**, 1027, doi: 10.1029/2001GC000252, 3.

Examples

```
data("plates")
head("plates")
```

plot_density *Circular Kernel Density Plot*

Description

Plots multiples of a von Mises, wrapped Cauchy, and wrapped Normal density distribution in a circular plot

Usage

```
plot_density(
  x,
  bw = NULL,
  kernel = c("vonmises", "wrappedcauchy", "wrappednormal"),
  weights = NULL,
  axial = TRUE,
  n = 512L,
  norm.density = TRUE,
  kappa = NULL,
  rho = NULL,
  sd = NULL,
  c = NULL,
  fill = FALSE,
  scale = 0,
  shrink = 1,
  add = TRUE,
  main = NULL,
  labels = TRUE,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE,
  grid = FALSE,
  ...
)
```

Arguments

x	Either an object of class "density" or a numeric vector of angles (in degrees) from which the estimate is to be computed
bw, kappa, rho, sd, c	numeric. Smoothing bandwidth expressed as the concentration parameter κ for the von Mises distribution, ρ for the wrapped Cauchy distribution, or σ for the wrapped normal distribution. Small and large values for the von Mises and wrapped normal/Cauchy distribution, respectively, gives smooth density lines. If not specified, parameter will be estimated using est.kappa() for the von Mises distribution, or set to $p\exp(-1)$ and 1 for the wrapped Cauchy and wrapped Normal distribution (where $p = 2$ when axial=TRUE and 1 otherwise), respectively.

kernel	character. The smoothing kernel to be used; one of "vonmises" (the default), "wrappedcauchy", "wrappednormal, for the von Mises, the wrapped Cauchy, and the wrapped Normal distribution.
weights	numeric. A vector of observation weights, of the same length as x, to give individual observations weight in the density estimate. Should sum to 1; a warning is issued if it doesn't (unless subdensity = TRUE). Defaults to equal weight 1/length(x) per observation, matching stats::density().
axial	Logical. Whether data are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
n	integer. Number of equally spaced angles at which the density is to be estimated.
norm.density	logical. Normalize the density?
fill	logical. Whether to fill the density curve or draw just a line (the default)
scale	numeric. radius of plotted circle. Default is 1.1.
shrink	numeric. parameter that controls the size of the plotted function. Default is 1.
add	logical. Add to existing plot? (TRUE by default).
main	a main title for the plot, see also title.
labels	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
at	Optional vector of angles at which tick marks should be plotted. Set at=numeric(0) to suppress tick marks.
cborder	logical. Whether to plot the primitive of the circular plot (TRUE by default) or not.
grid	logical. Whether a grid should be added.
...	Further graphical parameters may also be supplied as arguments.

Value

plot or calculated densities

See Also

circular_density()

Other circular-plot: plot_points(), rose(), rose_geom, rose_stats()

Examples

```
set.seed(20250411)
# Filled von Mises kernel density curve inside the plot
plot_density(san_andreas$azi,
  kappa = 100,
  fill = TRUE, col = "#51127C80", border = "#51127CFF",
  grid = TRUE,
  add = FALSE
)

# Superimpose a wrapped Cauchy kernel distribution curve
```

```
plot_density(san_andreas$azi,
  rho = 0.9, kernel = "wrappedcauchy",
  fill = FALSE, col = "#FB8861FF",
  add = TRUE
)

# Superimpose a wrapped Normal kernel distribution curve
plot_density(san_andreas$azi,
  sd = 2, kernel = "wrappednormal",
  fill = FALSE, col = "#E65164FF",
  add = TRUE
)

# Superimpose a von Mises kernel density curve on a rose diagram:
rose(san_andreas$azi, grid = TRUE)
plot_density(san_andreas$azi,
  bw = 100, col = "#51127CFF",
  add = TRUE, lwd = 3
)

# Corona plot (density curve outside of a rose diagram plot):
w <- weighting(san_andreas$unc)
rose(san_andreas$azi, weights = w, dots = TRUE, stack = TRUE, dot_cex = 0.5, dot_pch = 21)
plot_density(san_andreas$azi, weights = w,
  bw = 100,
  scale = 1.1, shrink = 3, xpd = NA,
  col = "#51127CFF"
)
```

`plot_points` *Add Points to a Circular Plot*

Description

Add points to a plot of circular data points on the current graphics device.

Usage

```
plot_points(
  x,
  axial = TRUE,
  stack = FALSE,
  binwidth = 1,
  cex = 1,
  sep = 0.025,
  jitter_factor = 0,
  ...,
  scale = 1.1,
  add = TRUE,
```

```
  main = NULL,
  labels = TRUE,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE
)
```

Arguments

x	Data to be plotted. A numeric vector containing angles (in degrees).
axial	Logical. Whether data are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
stack	logical: if TRUE, points are stacked on the perimeter of the circle. Otherwise, all points are plotted on the perimeter of the circle. Default is FALSE.
binwidth	numeric. Bin width (in degrees) for the stacked dot plots. ignored when stack==FALSE. Is set to 1 degree by default.
cex	character (or symbol) expansion: a numerical vector. This works as a multiple of par("cex").
sep	constant used to specify the distance between stacked points, if stack==TRUE or in the case of more than one dataset. Default is 0.025; smaller values will create smaller spaces.
jitter_factor	numeric. Adds a small amount of random variation to the location of each points along radius that is added to scale. Jitter is ignored when stack==TRUE). If 0, no jitter is added (by default); if negative, the points fall into the circle.
...	Further graphical parameters may also be supplied as arguments.
scale	numeric. Scales radius of plotted dots Default is 1.1.
add	logical. Add to existing plot? (TRUE by default).
main	a main title for the plot, see also title.
labels	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
at	Optional vector of angles at which tick marks should be plotted. Set at=numeric(0) to suppress tick marks.
cborder	logical. Whether to plot the primitive of the circular plot (TRUE by default) or not.

Value

A list with information on the plot

See Also

Other circular-plot: plot_density(), rose(), rose_geom, rose_stats()

Examples

```
set.seed(20250511)
x <- rvm(100, mean = 90, k = 5)

# plot poinit without jitter
plot_points(x, add = FALSE)

# with some jitter
plot_points(x, jitter_factor = .2, add = FALSE)

# stacked dots:
plot_points(x, stack = TRUE, binwidth = 3, add = FALSE, xpd = TRUE)
```

`PoR2Geo_azimuth` *Azimuth Conversion From PoR to Geographical Coordinate Reference System*

Description

Conversion of PoR azimuths into geographical azimuths

Usage

```
PoR2Geo_azimuth(x, PoR, axial = TRUE)
```

Arguments

`x`	`sf` object or a `data.frame` containing the coordinates of the point(s) (`lat`, `lon` columns). `x` must contain the direction of σ_{Hmax} as column `azi`, its standard deviation (column `unc`) is optional).
`PoR`	`data.frame` or object of class `euler.pole` containing the geographical coordinates of the Euler pole.
`axial`	logical. Whether the azimuth is axial (0-180°) or directional (0-360°).

Value

numeric vector of transformed azimuths (in degrees)

References

Stephan, T., Enkelmann, E., and Kroner, U. "Analyzing the horizontal orientation of the crustal stress adjacent to plate boundaries". *Sci Rep* 13. 15590 (2023). doi:10.1038/s41598023424332.

See Also

`PoR_shmax()`

Examples

```
data("nuvel1")
# North America relative to Pacific plate:
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
data("san_andreas")
san_andreas$azi.PoR <- PoR_shmax(san_andreas, PoR)

# convert back to geo CRS
PoR2Geo_azimuth(san_andreas, PoR) |> head()
```

PoR_azi *Azimuth Conversion from Geographical to PoR Coordinate Reference System*

Description

Transforms azimuths and models the direction of maximum horizontal stress σ_{Hmax} in the Euler pole (Pole of Rotation) coordinate reference system. When type of plate boundary is given, it also gives the deviation from the theoretically predicted azimuth of σ_{Hmax}, the circular distance, and the normalized χ^2 statistics.

Usage

```
PoR_azimuth(x, PoR, axial = TRUE)

PoR_shmax(x, PoR, type = c("none", "in", "out", "right", "left"), axial = TRUE)
```

Arguments

x	sf object or a data.frame containing the coordinates of the point(s) (lat, lon columns). x must contain the direction of σ_{Hmax} as column azi, its standard deviation (column unc) is optional).
PoR	data.frame or object of class euler.pole containing the geographical coordinates of the Euler pole.
axial	logical. Whether the azimuth is axial (0-180°) or directional (0-360°).
type	Character. Type of plate boundary (optional). Can be "out", "in", "right", or "left" for outward, inward, right-lateral, or left-lateral moving plate boundaries, respectively. If "none" (the default), only the PoR-equivalent azimuth is returned.

Details

The theoretical azimuth of σ_{Hmax} in the pole of rotation reference system is 0 (or 180), 45, 90, 135° if the stress is sourced by an outward, sinistral, inward, or dextral moving plate boundary, respectively. directions of σ_{Hmax} with respect to the four plate boundary types.

Value

PoR_azimuth returns numeric vector of the transformed azimuth in degrees. PoR_shmax returns either a numeric vector of the azimuths in the transformed coordinate system (in degrees), or a "data.frame" with

azi.PoR the transformed azimuths (in degrees),

prd the predicted azimuths (in degrees),

dev the deviation between the transformed and the predicted azimuth in degrees (positive for counterclockwise deviation of observed azimuth wrt. predicted azimuth),

nchisq the Norm χ^2 test statistic, and

cdist the angular distance between the transformed and the predicted azimuth.

References

Stephan, T., Enkelmann, E., and Kroner, U. "Analyzing the horizontal orientation of the crustal stress adjacent to plate boundaries". *Sci Rep* 13. 15590 (2023). doi:10.1038/s41598023424332.

See Also

model_shmax() to compute the theoretical direction of σ_{Hmax} in the geographical reference system. deviation_shmax() to compute the deviation of the modeled direction from the observed direction of σ_{Hmax}. norm_chisq() to calculate the normalized χ^2 statistics. circular_distance() to calculate the angular distance.

Examples

```
data("nuvel1")
# North America relative to Pacific plate:
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")

data("san_andreas")
res <- PoR_shmax(san_andreas, PoR, type = "right")
head(res)
```

PoR_coordinates *Coordinates of the Pole of Rotation Reference System*

Description

Retrieve the PoR equivalent coordinates of an object

Usage

```
PoR_coordinates(x, PoR)
```

Arguments

x	Can be either a "data.frame" containing lat and lon coordinates of a point in the geographical CRS or the lat.PoR, lon.PoR) of the point in the PoR CRS, a two-column matrix containing the lat and lon coordinates, a sf object, or a raster object.
PoR	Pole of Rotation. "data.frame" containing the geographical coordinates of the Euler pole

Value

PoR_coordinates() returns data.frame with the PoR coordinates (lat.PoR, lon.PoR).

Examples

```
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to Pacific plate
data("san_andreas")

# coordinates from sf object
san_andreas.por_sf <- PoR_coordinates(san_andreas, por)
head(san_andreas.por_sf)

# coordinates from data.frame
san_andreas.por_df <- PoR_coordinates(sf::st_drop_geometry(san_andreas), por)
head(san_andreas.por_df)
```

PoR_crs *PoR coordinate reference system*

Description

Create the reference system transformed in Euler pole coordinate

Usage

```
PoR_crs(x)
```

Arguments

x	"data.frame" or "euler.pole" object containing the geographical coordinates of the Euler pole

Details

The PoR coordinate reference system is oblique transformation of the geographical coordinate system with the Euler pole coordinates being the the translation factors.

Value

Object of class `crs`

See Also

`sf::st_crs()`

Examples

```
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to Pacific plate
PoR_crs(por)
```

`PoR_distance` *Distance to Pole of Rotation*

Description

Retrieve the (angular) great-circle distance between a point and the PoR (Euler pole).

Usage

```
PoR_distance(x, PoR, FUN = orthodrome)
```

Arguments

`x`	Can be either a `"data.frame"` containing `lat` and `lon` coordinates of a point in the geographical CRS or the `lat.PoR`, `lon.PoR`) of the point in the PoR CRS, a two-column matrix containing the lat and lon coordinates, a `sf` object, or a `raster` object.
`PoR`	Pole of Rotation. `"data.frame"` containing the geographical coordinates of the Euler pole
`FUN`	function to calculate the great-circle distance. `orthodrome()`, `haversine()` (the default), or `vincenty()`.

Value

numeric vector. Great-circle distance in degree

Examples

```
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to Pacific plate
data("san_andreas")

# distance form sf object
PoR_distance(san_andreas, por) |> head()
```

```
# distance form data.frame
PoR_distance(sf::st_drop_geometry(san_andreas), por) |> head()
PoR_distance(sf::st_drop_geometry(san_andreas), por, FUN = orthodrome) |> head()
PoR_distance(sf::st_drop_geometry(san_andreas), por, FUN = vincenty) |> head()
```

PoR_map *Map of data in Pole of Rotation reference frame*

Description

Transforms the spatial data and azimuths into the PoR reference frame and shows them in a map

Usage

```
PoR_map(
  x,
  PoR,
  pb = NULL,
  type = c("none", "in", "out", "right", "left"),
  show.deviation = FALSE,
  ...
)
```

Arguments

x, pb	sf objects of the data points and the plate boundary geometries in the geographical coordinate system
PoR	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Pole of Rotation
type	Character. Type of plate boundary (optional). Can be "out", "in", "right", or "left" for outward, inward, right-lateral, or left-lateral moving plate boundaries, respectively. If "none" (the default), only the PoR-equivalent azimuth is returned.
show.deviation	logical. Whether the data should be color-coded according to the deviation from the prediction, or according to the stress regime? Is ignored if type=='none'.
...	optional arguments passed to tectonicr.colors()

Value

plot

See Also

PoR_shmax(), axes(), tectonicr.colors()

Examples

```
data("nuvel1")
na_pa <- subset(nuvel1, nuvel1$plate.rot == "na")

data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")

data("san_andreas")
PoR_map(san_andreas, PoR = na_pa, pb = plate_boundary, type = "right", show.deviation = TRUE)
```

PoR_stress2grid *Spatial Interpolation of SHmax in PoR Coordinate Reference System*

Description

The data is transformed into the PoR system before the interpolation. The interpolation grid is returned in geographical coordinates and azimuths.

Usage

```
PoR_stress2grid(
  x,
  PoR,
  grid = NULL,
  PoR_grid = TRUE,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2.5,
  remove_PoR = FALSE,
  ...
)

PoR_stress2grid_stats(
  x,
  PoR,
  grid = NULL,
  PoR_grid = TRUE,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2.5,
  remove_PoR = FALSE,
  ...
)
```

Arguments

x	sf object containing

azi σ_{Hmax} in degree

unc Uncertainties of σ_{Hmax} in degree

type Methods used for the determination of the orientation of σ_{Hmax}

`PoR` Pole of Rotation. `data.frame` or object of class `"euler.pole"` containing the geographical coordinates of the Euler pole

`grid` (optional) Point object of class `sf`.

`PoR_grid` logical. Whether the grid should be generated based on the coordinate range in the PoR (`TRUE`, the default) CRS or the geographical CRS (`FALSE`). Is ignored if `grid` is specified.

`lon_range`, `lat_range` (optional) numeric vector specifying the minimum and maximum longitudes and latitudes (are ignored if `grid` is specified).

`gridsize` Numeric. Target spacing of the regular grid in decimal degree. Default is `2.5` (is ignored if `grid` is specified)

`remove_PoR` logical. Whether PoR azimuths and coordinates will be removed from final output or not (the default.)

`...` Arguments passed to `stress2grid()`

Details

Stress field and wavelength analysis in PoR system and back-transformed

Value

`sf` object containing

lon,lat longitude and latitude in geographical CRS (in degrees)

lon.PoR,lat.PoR longitude and latitude in PoR CRS (in degrees). Only if `remove_PoR=TRUE`

azi geographical mean σ_{Hmax} in degree

azi.PoR PoR mean σ_{Hmax} in degree. Only if `remove_PoR=TRUE`

sd Standard deviation of σ_{Hmax} in degrees

R Search radius in km

mdr Mean distance of datapoints per search radius

N Number of data points in search radius

See Also

`stress2grid()`, `compact_grid()`

Examples

```
data("san_andreas")
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
PoR_stress2grid(san_andreas, PoR) |> head()
```

```
## Not run:
PoR_stress2grid_stats(san_andreas, PoR, mode = TRUE) |> head()

## End(Not run)
```

`por_transformation` *Conversion between spherical PoR to geographical coordinate system*

Description

Transformation from spherical PoR to geographical coordinate system and vice versa

Usage

```
geographical_to_PoR(x, PoR)

PoR_to_geographical(x, PoR)
```

Arguments

`x`	Can be either a `"data.frame"` containing `lat` and `lon` coordinates of a point in the geographical CRS or the `lat.PoR`, `lon.PoR`) of the point in the PoR CRS, a two-column matrix containing the lat and lon coordinates, a `sf` object, or a `raster` object.
`PoR`	Pole of Rotation. `"data.frame"` containing the geographical coordinates of the Euler pole

Value

object of same type of `x` with the transformed coordinates. If `x` was a `data.frame`, transformed coordinates are named `lat.PoR` and `lon.PoR` for PoR CRS, or `lat` and `lon` for geographical CRS).

Examples

```
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to Pacific plate
data("san_andreas")
san_andreas.por <- geographical_to_PoR(san_andreas, por)
head(san_andreas.por)
head(PoR_to_geographical(san_andreas.por, por))
```

prd_err *Error of Model's Prediction*

Description

The maximum error in the model's predicted azimuth given the Pole of rotations uncertainty and distance of the data point to the pole.

Usage

```
prd_err(dist_PoR, sigma_PoR = 1)
```

Arguments

dist_PoR	Distance to Euler pole (great circle distance, in degree)
sigma_PoR	uncertainty of the position of the Pole of rotation (in degree).

Value

numeric vector. The maximum error for azimuths prediction (in degree)

References

Ramsay, J.A. Folding and fracturing of rocks. McGraw-Hill, New York, 1967.

See Also

PoR_shmax() and model_shmax() for the model's prediction, and orthodrome() for great circle distances.

Examples

```
prd_err(67, 1)

# San Andreas example:
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to Pacific plate
data("san_andreas")
d <- PoR_distance(san_andreas, por)
prd_err(d)
```

`projected_pb_strike` *Strike of the plate boundary projected on data point*

Description

The fault's strike in the PoR CRS projected on the data point along the predicted stress trajectories.

Usage

```
projected_pb_strike(x, PoR, pb, tangential = FALSE, ...)
```

Arguments

x, pb	`sf` objects of the data points and the plate boundary geometries in the geographical coordinate system
PoR	Pole of Rotation. `"data.frame"` or object of class `"euler.pole"` containing the geographical coordinates of the Pole of Rotation
tangential	Logical. Whether the plate boundary is a tangential boundary (`TRUE`) or an inward and outward boundary (`FALSE`, the default).
...	optional arguments passed to `smoothr::densify()`

Details

Useful to calculate the beta angle, i.e. the angle between σ_{Hmax} direction (in PoR CRS!) and the fault's strike (in PoR CRS). The beta angle is the same in geographical and PoR coordinates.

Value

Numeric vector of the strike direction of the plate boundary (in degree)

Note

The algorithm calculates the great circle bearing between line vertices. Since transform plate boundaries represent small circle lines in the PoR system, this great-circle azimuth is only a approximation of the true (small-circle) azimuth.

Examples

```
data("nuvel1")
na_pa <- subset(nuvel1, nuvel1$plate.rot == "na")

data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")

data("san_andreas")
res <- projected_pb_strike(
  x = san_andreas, PoR = na_pa, pb = plate_boundary, tangential = TRUE
)
```

```
head(res)
head(san_andreas$azi - res) # beta angle
```

`quick_plot` *Plotting Stress Analysis Results*

Description

Creates a set of plots including the azimuth as a function of the distance to the plate boundary, the Norm Chi-squared as a function of the distance to the plate boundary, the circular distance (and dispersion) a function of the distance to the plate boundary, a von-Mises QQ plot, and a rose diagram of the quality-weighted frequency distribution of the azimuths.

Usage

```
quick_plot(azi, distance, prd, unc = NULL, regime, width = 51)
```

Arguments

`azi`	numeric. Azimuth of σ_{Hmax}
`distance`	numeric. Distance to plate boundary
`prd`	numeric. the predicted direction of σ_{Hmax}
`unc`	numeric. Uncertainty of observed σ_{Hmax}, either a numeric vector or a number
`regime`	character vector. The stress regime (following the classification of the World Stress Map)
`width`	integer. window width (in number of observations) for moving average of the azimuths, circular dispersion, and Norm Chi-square statistics. If `NULL`, an optimal width will be estimated.

Details

Plot 1 shows the transformed azimuths as a function of the distance to the plate boundary. The red line indicates the rolling circular mean, stippled red lines indicate the 95% confidence interval about the mean.

Plot 2 shows the normalized χ^2 statistics as a function of the distance to the plate boundary. The red line shows the rolling χ^2 statistic.

Plot 3 shows the circular distance of the transformed azimuths to the predicted azimuth, as a function of the distance to the plate boundary. The red line shows the rolling circular dispersion about the prediction.

Plot 4 give the rose diagram of the transformed azimuths.

Value

four R base plots

See Also

PoR_shmax(), distance_from_pb(), circular_mean(), circular_dispersion(), confidence_interval_fisher(), norm_chisq(), weighted_rayleigh(), vm_qqplot()

Examples

```
data("nuvel1")
na_pa <- subset(nuvel1, nuvel1$plate.rot == "na")

data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")

data("san_andreas")
res <- PoR_shmax(san_andreas, na_pa, "right")
d <- distance_from_pb(san_andreas, na_pa, plate_boundary, tangential = TRUE)
quick_plot(res$azi.PoR,
  distance = d, prd = res$prd, unc = san_andreas$unc,
  regime = san_andreas$regime
)
```

---

rao_spacing_test *Rao Spacing Test*

Description

Rao's spacing test of uniformity on the circle

Usage

```
rao_spacing_test(x, alpha = 0.05, n_perm = 9999L, axial = TRUE)
```

Arguments

x	numeric vector. Values in degrees
alpha	Significance level of the test ranging between 0 and 1.
n_perm	integer. Number of simulated uniform samples.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.

Details

Test statistic is

$$L = 1/2 \sum(|T_i - 360/n|)$$

in degrees, with the spacings T_i between consecutive angles (Mardia & Jupp 2000, eq. 6.3.45). The p-value is the proportion of n_perm simulated uniform samples with a larger L

Value

A list with class "htest" containing the following components:

statistic the value of the L statistic

parameter the number of data and the number of simulations

p.value the p-value for the test

alternative a character string describing the alternative hypothesis.

method a character string indicating what type of test was performed.

data.name a character string giving the name(s) of the data.

See Also

Other One-Sample-Tests: hodges_ajne_test(), kuiper_test(), norm_chisq(), rayleigh-test, watson_test(), weighted-rayleigh-test

Examples

```
set.seed(20250411)
rao_spacing_test(rvm(100, mean = 0, kappa = 10))

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
rao_spacing_test(sa.por$azi.PoR, alpha = .05)
```

---

rayleigh-test *Rayleigh Test of Circular Uniformity*

Description

A test to determine whether a sample of circular or directional data is evenly spread out or clustered around a single specific direction. The test assesses the significance of the mean resultant length. rayleigh_test_perm() estimates p-values by Monte Carlo simulation of uniform samples.

Usage

```
rayleigh_test(x, mu = NULL, axial = TRUE, alpha = 0.05, quiet = FALSE)

rayleigh_test_perm(x, mu = NULL, axial = TRUE, n_perm = 1000L)
```

Arguments

x	numeric vector. Values in degrees
mu	(optional) The specified or known mean direction (in degrees) in alternative hypothesis.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.
alpha	Significance level of the test ranging between 0 and 1.
quiet	logical. Prints the test's decision.
n_perm	integer. Number of simulated uniform samples.

Details

Hypotheses:

Null Hypothesis H_0: The population is distributed uniformly (randomly) around the circle with no preferred direction.

Alternative Hypothesis H_1: The population is not uniform and has a unimodal (single-peaked) concentration in a preferred direction. When mu is specified (V-test), the alternative is a unimodal distribution centred on the specified direction; the test is one-sided.

Mean Resultant Length ($\bar{R}$ or R): A value between 0 and 1 that measures how concentrated the data points are.

- $\bar{R}$ = 0: The data is completely spread out around the circle
- $\bar{R}$ = 1: All data points point in the exact same direction.

p-value The probability of seeing data this clustered purely by chance under the assumption of uniformity.

Interpretation:

- Small p-value (p < 0.05): Reject the null hypothesis. The length of the mean resultant differs significantly from zero, and the angles are not randomly distributed. You have strong evidence that the data points share a significant preferred or mean direction (unimodal clustering).
- Large p-value (p $\geq$ 0.05): Fail to reject the null hypothesis. There is not enough evidence to claim a preferred direction, meaning the data looks random or uniform around the circle.

Value

a list with the components:

R **or** C mean resultant length or, if mu is specified, the mean cosine of the deviations from mu. Large values of R (large positive values of C) reject uniformity. Negative values of C indicate that the vectors point away from mu; the one-sided V-test does not reject them.

statistic test statistic

p.value significance level of the test statistic

Note

Although the Rayleigh test is consistent against (non-uniform) von Mises alternatives, it is not consistent against alternatives with a mean resultant length $\rho = 0$ (in particular, distributions with antipodal symmetry, i.e. undoubled axial data). Tests of non-uniformity which are consistent against all alternatives include Kuiper's test (`kuiper_test()`) and Watson's U^2 test (`watson_test()`).

Limitations:

- The test assumes a unimodal alternative (one main peak).
- If your data has two opposite clusters (bimodal or axial data, like a bi-directional line trend), the Rayleigh test can yield a high/non-significant p-value because the opposing vectors cancel each other out.

References

Fisher, N. I. (1993) Statistical Analysis of Circular Data, Cambridge University Press.

See Also

`mean_resultant_length()`, `circular_mean()`

Other One-Sample-Tests: `hodges_ajne_test()`, `kuiper_test()`, `norm_chisq()`, `rao_spacing_test()`, `watson_test()`, `weighted-rayleigh-test`

Examples

```
# Example data from Mardia and Jupp (1999), pp. 93
rayleigh_test(homing, axial = FALSE) # Do not reject null hypothesis.
# R = 0.22; stat = 0.497, p = 0.62
rayleigh_test_perm(homing, axial = FALSE)

# Example data from Davis (1986), pp. 316
rayleigh_test(striae, axial = FALSE) # reject null hypothesis
rayleigh_test_perm(striae, axial = FALSE) # reject null hypothesis

rayleigh_test(striae, mu = 105, axial = FALSE) # reject null hypothesis
rayleigh_test_perm(striae, mu = 105, axial = FALSE) # reject null hypothesis

# Example data from Mardia and Jupp (1999), pp. 99
atomic_weight <- c(
  rep(0, 12), rep(3.6, 1), rep(36, 6), rep(72, 1),
  rep(108, 2), rep(169.2, 1), rep(324, 1)
)
rayleigh_test(atomic_weight, 0, axial = FALSE) # reject null hypothesis

# San Andreas Fault Data:
data(san_andreas)
rayleigh_test(san_andreas$azi) # reject null hypothesis
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
rayleigh_test(sa.por$azi.PoR, mu = 135) # reject null hypothesis
rayleigh_test_perm(sa.por$azi.PoR, mu = 135, n_perm = 1e3) # reject null hypothesis
```

`relative_rotation` *Relative rotation between two rotations*

Description

Calculates the relative rotation between two rotations, i.e. the difference from rotation 1 to rotation 2.

Usage

```
relative_rotation(r1, r2)
```

Arguments

`r1`, `r2`	Objects of class `"euler.pole"`. First rotation is `r1`, followed rotation `r2`.

Value

`list`. Euler axes (geographical coordinates) and Euler angles (in degrees)

References

Schaeben, H., Kroner, U. and Stephan, T. (2021). Euler Poles of Tectonic Plates. In B. S. Daza Sagar, Q. Cheng, J. McKinley and F. Agterberg (Eds.), *Encyclopedia of Mathematical Geosciences. Encyclopedia of Earth Sciences Series* (pp. 1–7). Springer Nature Switzerland AG 2021. doi: 10.1007/978-3-030-26050-7_435-1.

See Also

`euler_pole()` for class `"euler.pole"`

Examples

```
a <- euler_pole(90, 0, angle = 45)
b <- euler_pole(0, 0, 1, geo = FALSE, angle = -15)
relative_rotation(a, b) # axis: -90, -180; angle: 60
relative_rotation(b, a) # axis: 90, 0; angle: 60
```

rolling_test *Apply Rolling Functions using Circular Statistical Tests for Uniformity*

Description

A generic function for applying a function to rolling margins of an array.

Usage

```
roll_normchisq(
  obs,
  prd,
  unc = NULL,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)

roll_rayleigh(
  obs,
  prd,
  unc = NULL,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)

roll_dispersion(
  x,
  y,
  w = NULL,
  w.y = NULL,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)

roll_confidence(
  x,
  conf.level = 0.95,
  w = NULL,
```

```
  axial = TRUE,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)

roll_dispersion_CI(
  x,
  y,
  w = NULL,
  w.y = NULL,
  R,
  conf.level = 0.95,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)

roll_dispersion_sde(
  x,
  y,
  w = NULL,
  w.y = NULL,
  R,
  conf.level = 0.95,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)
```

Arguments

obs	Numeric vector containing the observed azimuth of σ_{Hmax}, same length as prd
prd	Numeric vector containing the modeled azimuths of σ_{Hmax}, i.e. the return object from model_shmax()
unc	Uncertainty of observed σ_{Hmax}, either a numeric vector or a number. Observations with missing uncertainty are dropped; an uncertainty of 0 is replaced by 1.
width	integer specifying the window width (in numbers of observations) which is aligned to the original sample according to the align argument. If NULL, an optimal width is estimated.
by.column	logical. If TRUE, FUN is applied to each column separately.

partial	logical or numeric. If FALSE then FUN is only applied when all indexes of the rolling window are within the observed time range. If TRUE (default), then the subset of indexes that are in range are passed to FUN. A numeric argument to partial can be used to determine the minimal window size for partial computations. See below for more details.
fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of zoo::na.fill() for details
...	optional arguments passed to zoo::rollapply()
x, y	numeric. Directions in degrees
w, w.y	(optional) Weights of x and y, respectively. A vector of positive numbers and of the same length as x.
conf.level	Level of confidence: $(1 - \alpha\%)/100$. (0.95 by default).
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
R	The number of bootstrap replicates.

Value

numeric vector with the test statistic of the rolling test. roll_dispersion_CI returns a 2-column matrix with the lower and the upper confidence limits

Note

If the rolling functions are applied to values that are a function of distance it is recommended to sort the values first.

Examples

```
data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
distance <- distance_from_pb(
  x = san_andreas,
  PoR = PoR,
  pb = plate_boundary,
  tangential = TRUE
)
dat <- san_andreas[order(distance), ]
dat.PoR <- PoR_shmax(san_andreas, PoR, "right")
roll_normchisq(dat.PoR$azi.PoR, 135, dat$unc) |> head()
roll_rayleigh(dat.PoR$azi.PoR, prd = 135, unc = dat$unc) |> head()
roll_dispersion(dat.PoR$azi.PoR, y = 135, w = 1 / dat$unc) |> head()
roll_confidence(dat.PoR$azi.PoR, w = 1 / dat$unc) |> head()

roll_dispersion_CI(dat.PoR$azi.PoR, y = 135, w = 1 / dat$unc, R = 10) |> head()
```

roll_circstats *Apply Rolling Functions using Circular Statistics*

Description

A generic function for applying a function to rolling margins of an array.

Usage

```
roll_circstats(
  x,
  w = NULL,
  FUN,
  axial = TRUE,
  na.rm = TRUE,
  width = NULL,
  by.column = FALSE,
  partial = TRUE,
  fill = NA,
  ...
)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
FUN	the function to be applied
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.
width	integer specifying the window width (in numbers of observations) which is aligned to the original sample according to the align argument. If NULL, an optimal width is calculated.
by.column	logical. If TRUE, FUN is applied to each column separately.
partial	logical or numeric. If FALSE then FUN is only applied when all indexes of the rolling window are within the observed time range. If TRUE (default), then the subset of indexes that are in range are passed to FUN. A numeric argument to partial can be used to determine the minimal window size for partial computations. See below for more details.
fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of zoo::na.fill() for details
...	optional arguments passed to zoo::rollapply()

Value

numeric vector with the results of the rolling function.

Note

If the rolling statistics are applied to values that are a function of distance it is recommended to sort the values first.

Examples

```
data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
distance <- distance_from_pb(
  x = san_andreas,
  PoR = PoR,
  pb = plate_boundary,
  tangential = TRUE
)
dat <- san_andreas[order(distance), ]
roll_circstats(dat$azi, w = 1 / dat$unc, circular_mean, width = 51) |> head()
```

---

rose *Rose Diagram*

Description

Plots a rose diagram (rose of directions), the analogue of a histogram or density plot for angular data.

Usage

```
rose(
  x,
  weights = NULL,
  binwidth = NULL,
  bins = NULL,
  axial = TRUE,
  equal_area = TRUE,
  muci = TRUE,
  origin.text = "N",
  main = NULL,
  sub = NULL,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE,
  labels = TRUE,
  col = "grey",
```

```
  dots = FALSE,
  dot_pch = 1,
  dot_cex = 1,
  dot_col = "slategrey",
  stack = FALSE,
  jitter_factor = 0,
  grid = FALSE,
  grid.lines = seq(0, 135, 45),
  grid.circles = seq(0.2, 1, 0.2),
  add = FALSE,
  rule = c("sturges", "rice"),
  ...
)
```

Arguments

x	Data to be plotted. A numeric vector containing angles (in degrees).
weights	Optional vector of numeric weights associated with x.
binwidth	The width of the bins (in degrees).
bins	number of arcs to partition the circle width. Overridden by binwidth.
axial	Logical. Whether data are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
equal_area	Logical. Whether the radii of the bins are proportional to the frequencies (equal_area=FALSE, i.e. equal-angle) or proportional to the square-root of the frequencies (equal_area=TRUE, the default).
muci	logical. Whether the mean and its 95% CI are added to the plot or not.
origin.text	character. String to be drawn at the top margin of the plot ("N" by default)
main, sub	Character string specifying the title and subtitle of the plot. If sub = NULL, it will show the bin width.
at	Optional vector of angles at which tick marks should be plotted. Set at=numeric(0) to suppress tick marks.
cborder	logical. Whether to plot the primitive of the circular plot (TRUE by default) or not.
labels	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
col	fill color of rose petals / fans
dots	logical. Whether a circular dot plot should be added (FALSE is the default).
dot_cex, dot_pch, dot_col	Plotting arguments for circular dot plot
stack	logical. Groups and stacks the dots if TRUE. Default is FALSE.
jitter_factor	Add a small amount of noise to the angles' radius that is added to scale. Jitter is ignored when stack==TRUE). If 0, no jitter is added (by default); if negative, the points fall into the circle.
grid	logical. Whether to add a grid to the circular plot. Default is FALSE.

grid.lines, grid.circles
: numeric. Adds a sequence of straight grid lines and circles based on angles (in degrees) and radii, respectively. Ignored when grid=FALSE

add
: logical. Add to existing plot? (TRUE by default).

rule
: character. Rule to select optimal bin width, one of "sturges" (the default) or "rice". The optimal number according to Sturges' rule is $1 + \log 2(n)$, while for Rice's rule it is $2n^{1/3}$ (less conservative for large n).

...
: Additional graphical arguments passed to graphics::polygon() for plotting the rose petals / fans

Value

class of histogram that is a list of the calculated frequencies parameters including the bin width.

See Also

Other circular-plot: plot_density(), plot_points(), rose_geom, rose_stats()

Examples

```
x <- rvm(100, mean = 90, k = 5)
rose(x, axial = FALSE, border = TRUE, grid = TRUE)

data("san_andreas")
rose(san_andreas$azi, main = "equal area")
rose(san_andreas$azi, equal_area = FALSE, main = "equal angle")

# weighted frequencies:
rose(san_andreas$azi, weights = 1 / san_andreas$unc, main = "weighted")

# add dots:
rose(san_andreas$azi, dots = TRUE, main = "dot plot", jitter = .2)

# stack dots:
rose(san_andreas$azi,
  dots = TRUE, stack = TRUE, dot_cex = 0.5, dot_pch = 21,
  main = "stacked dot plot"
)
```

---

rose_geom *Direction Lines and Fans in Circular Diagram*

Description

Direction Lines and Fans in Circular Diagram

Usage

```
rose_line(x, radius = 1, axial = TRUE, add = TRUE, ...)

rose_fan(x, d, radius = 1, axial = TRUE, add = TRUE, ...)
```

Arguments

x	angles in degrees
radius	of the plotted circle
axial	Logical. Whether x are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
add	logical. Add to existing plot?
...	optional arguments passed to graphics::segments() or graphics::polygon()
d	width of a fan (in degrees)

Value

No return value, called for side effects

See Also

Other circular-plot: plot_density(), plot_points(), rose(), rose_stats()

Examples

```
angles <- c(0, 10, 45)
radius <- c(.7, 1, .2)
lwd <- c(2, 1, .75)
col <- c(1, 2, 3)
rose_line(angles, radius = radius, axial = FALSE, add = FALSE, lwd = lwd, col = col)
```

---

rose_stats *Show Average Direction and Spread in Rose Diagram*

Description

Adds the average direction (and its spread) to an existing rose diagram.

Usage

```
rose_stats(
  x,
  weights = NULL,
  axial = TRUE,
  avg = c("mean", "median", "sample_median"),
  spread = c("CI", "fisher", "sd", "IQR", "mdev"),
  avg.col = "#B63679FF",
```

```
  avg.lty = 2,
  avg.lwd = 1.5,
  spread.col = ggplot2::alpha("#B63679FF", 0.2),
  spread.border = FALSE,
  spread.lty = NULL,
  spread.lwd = NULL,
  add = TRUE,
  ...
)
```

Arguments

x	Data to be plotted. A numeric vector containing angles (in degrees).
weights	Optional vector of numeric weights associated with x.
axial	Logical. Whether data are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
avg	character. The average estimate for x. Either the circular mean ("mean", the default), the circular Quasi Median ("median"), or the sample median ("sample_median").
spread	character. The measure of spread to be plotted as a fan. Either Batchelet's 95% confidence interval by ("CI", the default), Fisher's 95% confidence interval ("fisher"), the circular standard deviation ("sd"), the Quasi interquartile range on the circle ("IQR"), or the sample median deviation ("mdev"). NULL if no fan should be drawn.
avg.col	color for the average line
avg.lty	line type of the average line
avg.lwd	line width of the average line
spread.col	color of the spread fan
spread.border	logical. Whether to draw a border of the fan or not.
spread.lty	line type of the spread fan's border
spread.lwd	line width of the spread fan's border
add	logical. Add to existing plot? (TRUE by default).
...	optional arguments to circular_plot() if add is FALSE.

Value

plot or a two-element vector containing the calculated average and spread when assigned.

See Also

circular_mean(), circular_median(), circular_sample_median(), confidence_interval(), confidence_interval_fisher(), circular_sd(), circular_IQR(), circular_sample_median_deviation() for statistical parameters.

Other circular-plot: plot_density(), plot_points(), rose(), rose_geom

Examples

```
data("san_andreas")
rose(san_andreas$azi, weights = 1 / san_andreas$unc, muci = FALSE)
rose_stats(san_andreas$azi, weights = 1 / san_andreas$unc, avg = "sample_median", spread = "mdev")
```

`sample_dispersion` *Sample circular dispersion*

Description

Alternative versions of variance, dispersion a distance (Mardia and Jupp, 1999; pp. 19-20). These alternative dispersion has a minimum at the sample median.

Usage

```
sample_circular_variance(x, w = NULL, axial = TRUE)

sample_circular_distance(x, y, axial = TRUE, na.rm = TRUE)

sample_circular_dispersion(
  x,
  y = NULL,
  w = NULL,
  w.y = NULL,
  axial = TRUE,
  na.rm = TRUE
)
```

Arguments

`x, y`	vectors of numeric values in degrees. `length(y)` is either `1` or `length(x)`
`w, w.y`	(optional) Weights. A vector of positive numbers and of the same length as `x`. `w.y` is the (optional) weight of `y`.
`axial`	logical. Whether the data are axial, i.e. pi-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`).
`na.rm`	logical value indicating whether `NA` values in `x` should be stripped before the computation proceeds.

References

N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.

Mardia, K.V., and Jupp, P.E (1999). Directional Statistics, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979

Examples

```
a <- c(0, 2, 359, 6, 354)
sample_circular_distance(a, 10) # distance to single value

b <- a + 90
sample_circular_distance(a, b) # distance to multiple values

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
sample_circular_variance(sa.por$azi.PoR)
sample_circular_dispersion(sa.por$azi.PoR, y = 135)
sample_circular_dispersion(sa.por$azi.PoR, y = 135, w = weighting(san_andreas$unc))
```

---

sample_median *Sample Circular Median and Deviation*

Description

Sample median direction for a vector of circular data

Usage

```
circular_sample_median(x, axial = TRUE, na.rm = TRUE)

circular_sample_median_deviation(x, axial = TRUE, na.rm = TRUE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Value

numeric

References

N.I. Fisher (1993) Statistical Analysis of Circular Data, Cambridge University Press.

Examples

```
set.seed(1)
x <- rvm(n = 100, mean = 0, kappa = 10)
circular_sample_median(x)
circular_sample_median_deviation(x)

data("san_andreas")
circular_sample_median(san_andreas$azi)
circular_sample_median_deviation(san_andreas$azi)
```

---

second_central_moment *Second Central Momentum*

Description

Measures the skewness (a measure of the asymmetry of the probability distribution) and the kurtosis (measure of the "tailedness" of the probability distribution). Standardized versions are the skewness and kurtosis normalized by the mean resultant length (Mardia 1972).

Usage

```
second_central_moment(x, w = NULL, axial = TRUE, na.rm = FALSE)
```

Arguments

x	numeric vector. Values in degrees, for which the mean, median or standard deviation are required.
w	(optional) Weights. A vector of positive numbers, of the same length as x.
axial	logical. Whether the data are axial, i.e. pi-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.

Details

Negative values of skewness indicate skewed data in counterclockwise direction.

Large kurtosis values indicate tailed, values close to 0 indicate packed data.

Value

list containing

skewness second central sine momentum, i.e. the skewness

std_skewness standardized skewness

kurtosis second central cosine momentum, i.e. the kurtosis

std_kurtosis standardized kurtosis

Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
second_central_moment(sa.por$azi.PoR)
second_central_moment(sa.por$azi.PoR, w = weighting(san_andreas$unc))
```

---

shortest_distance_to_line

Shortest distance between pairs of geometries

Description

The shortest Great Circle distance between pairs of geometries

Usage

```
shortest_distance_to_line(x, line, ellipsoidal = FALSE)
```

Arguments

x, line	objects of class sfg, sfc or sf
ellipsoidal	Logical. Whether the distance is calculated using spherical distances (sf::st_distance()) or ellipsoidal distances (lwgeom::st_geod_distance()).

Value

numeric. Shortest distance in meters

Examples

```
plate_boundary <- subset(plates, plates$pair == "na-pa")
shortest_distance_to_line(san_andreas, plate_boundary) |>
  head()
```

---

spec_atan

Quadrant-specific inverse of the tangent

Description

Returns the quadrant specific inverse of the tangent

Usage

```
atan2_spec(x, y)

atan2d_spec(x, y)
```

Arguments

x, y	dividend and divisor that comprise the sum of sines and cosines, respectively.

Value

numeric.

References

Jammalamadaka, S. Rao, and Ambar Sengupta (2001). Topics in circular statistics. Vol. 5. world scientific.

---

spherical_angle *Angle along great circle on spherical surface*

Description

Smallest angle between two points on the surface of a sphere, measured along the surface of the sphere

Usage

```
orthodrome(lat1, lon1, lat2, lon2)

haversine(lat1, lon1, lat2, lon2)

vincenty(lat1, lon1, lat2, lon2)
```

Arguments

lat1, lat2	numeric vector. latitudes of point 1 and 2 (in radians)
lon1, lon2	numeric vector. longitudes of point 1 and 2 (in radians)

Details

`"orthodrome"` based on the spherical law of cosines

`"haversine"` uses haversine formula that is optimized for 64-bit floating-point numbers

`"vincenty"` uses Vincenty formula for an ellipsoid with equal major and minor axes

Value

numeric. Angle in radians

References

Imboden, C. & Imboden, D. (1972). Formel fuer Orthodrome und Loxodrome bei der Berechnung von Richtung und Distanz zwischen Beringungs- und Wiederfundort. *Die Vogelwarte* **26**, 336-346.

Sinnott, Roger W. (1984). Virtues of the Haversine. *Sky and telescope* **68**(2), 158.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, **23**(176), 88<U+2013>93. doi:10.1179/sre.1975.23.176.88.

http://www.movable-type.co.uk/scripts/latlong.html

http://www.edwilliams.org/avform147.htm

Examples

```
berlin <- c(52.52, 13.41) |> deg2rad()
calgary <- c(51.04, -114.072) |> deg2rad()
orthodrome(berlin[1], berlin[2], calgary[1], calgary[2]) # 1.176406
haversine(berlin[1], berlin[2], calgary[1], calgary[2]) # 1.176406
vincenty(berlin[1], berlin[2], calgary[1], calgary[2]) # 1.176406
```

---

stress2grid *Spatial Interpolation of SHmax*

Description

Stress field interpolation and wavelength analysis using a kernel (weighted) mean/median and standard deviation/IQR of stress data. Parameters can be adjusted to have inverse-distance-weighting (IDW) or nearest-neighbor interpolations (NN).

Usage

```
stress2grid(
  x,
  stat = c("mean", "median", "tensor"),
  grid = NULL,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2,
  min_data = 3L,
  max_data = Inf,
  max_sd = Inf,
  threshold = deprecated(),
  min_dist_threshold = 200,
  arte_thres = deprecated(),
  method_weighting = FALSE,
  quality_weighting = TRUE,
  dist_weighting = c("inverse", "linear", "none"),
  idp = 1,
  qp = 1,
```

```
  mp = 1,
  dist_threshold = 0.1,
  R_range = seq(50, 1000, 50),
  axial = TRUE
)

stress2grid_stats(
  x,
  grid = NULL,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2,
  min_data = 4L,
  max_data = Inf,
  threshold = deprecated(),
  min_dist_threshold = 200,
  arte_thres = deprecated(),
  method_weighting = FALSE,
  quality_weighting = TRUE,
  dist_weighting = c("inverse", "linear", "none"),
  idp = 1,
  qp = 1,
  mp = 1,
  dist_threshold = 0.1,
  R_range = seq(50, 1000, 50),
  mode = FALSE,
  kappa = 10,
  axial = TRUE,
  ...
)
```

Arguments

x	sf object containing **azi** σ_{Hmax} in degree **unc** (optional) Uncertainties of SHmax in degree **type** (optional) Methods used for the determination of the direction of σ_{Hmax}
stat	whether the direction of interpolated σ_{Hmax} is based on the circular mean and standard deviation ("mean", the default), the quasi-circular median and quasi-interquartile range ("median"), or the orientation tensor based principal direction and dispersion ("tensor").
grid	(optional) Point object of class sf.
lon_range, lat_range	(optional) numeric vector specifying the minimum and maximum longitudes and latitudes (ignored if grid is specified).
gridsize	numeric. Target spacing of the regular grid in decimal degree. Default is 2.5. (is ignored if grid is specified)

`min_data` integer. If the number of observations within distance `R_range` is less than `min_data`, a missing value `NA` will be generated. Default is `3` for `stress2grid()` and `4` for `stress2grid_stats()`.

`max_data` integer. The number of nearest observations that should be used for prediction, where "nearest" is defined in terms of the space of the spatial locations. Default is `Inf`.

`max_sd` numeric. Threshold for deviation of direction in degrees; if exceeds, missing values will be generated.

`threshold` **[Deprecated]** is no longer supported; use `max_sd` instead.

`min_dist_threshold` numeric. Distance threshold for smallest distance of the prediction location to the next observation location. Default is `200` km.

`arte_thres` **[Deprecated]** is no longer supported; use `min_dist_threshold` instead.

`method_weighting` logical. If a method weighting should be applied: Default is `FALSE`. If `FALSE`, overwrites `mp`.

`quality_weighting` logical. If a quality weighting should be applied: Default is `TRUE`. If `FALSE`, overwrites `qp`.

`dist_weighting` Distance weighting method which should be used. One of `"none"`, `"linear"`, or `"inverse"` (the default).

`idp, qp, mp` numeric. The weighting power of inverse distance, quality and method (the higher the value, the more weight). Default is `1`. When set to `0`, no weighting is applied. Only effective when `dist_weighting=="inverse"`.

`dist_threshold` numeric. Distance weight to prevent overweight of data nearby (0 to 1). Default is `0.1`

`R_range` numeric value or vector specifying the kernel half-width(s) search radii, i.e. the maximum distance from the prediction location to be used for prediction (in km). Default is `seq(50, 1000, 50)`. If combined with `max_data`, both criteria apply.

`axial` Logical. Whether angles in `x$azi` are uniaxial (`FALSE`) or biaxial (`TRUE`, the default).

`mode` logical. Should the circular mode be included in the statistical summary (slow)?

`kappa` numeric. von Mises distribution concentration parameter used for the circular mode. Will be estimated using `est.kappa()` if not provided.

`...` (optional) arguments to `circular_summary()`

Details

`stress2grid()` is originally based on the MATLAB script "stress2grid" by Ziegler and Heidbach (2019): https://github.com/MorZieg/Stress2Grid. The tectonicr version has been significantly modified to provide better performance and more flexibility.

`stress2grid_stats()` is based on `stress2grid()` but calculates circular summary statistics (see `circular_summary()`).

Value

`sf` object containing

lon,lat longitude and latitude in degrees

azi Circular mean od median SHmax in degree

sd Circular standard deviation or Quasi-IQR on the Circle of σ_{Hmax} in degrees

R Search radius in km

mdr Mean distance between grid point and datapoints per search radius

N Number of data points in search radius

When `stress2grid_stats()`, `azi` and `sd` are replaced by the output of `circular_summary()`.

Note

Although specialized for stress fields, this spatial interpolation algorithm can be applied to type of data. Please adjust the `axial` argument when applied to different, directional data sets.

References

Ziegler, M. and Heidbach, O. (2019). Matlab Script Stress2Grid v1.1. GFZ Data Services. doi:10.5880/wsm.2019.002

See Also

`dist_greatcircle()`, `PoR_stress2grid()`, `compact_grid()`, `circular_mean()`, `circular_median()`, `circular_sd()`, `circular_summary()`

Examples

```
data("san_andreas")

# Inverse Distance Weighting interpolation:
stress2grid(san_andreas, stat = "median") |> head()

stress2grid(san_andreas, stat = "tensor") |> head()

# Nearest Neighbor interpolation:
stress2grid(san_andreas, stat = "median", max_data = 5) |> head()

## Not run:
stress2grid_stats(san_andreas, mode = TRUE) |> head()

## End(Not run)
```

`stress_analysis` *Quick analysis of a stress data set*

Description

Returns the converted azimuths, distances to the plate boundary, statistics of the model, and some plots.

Usage

```
stress_analysis(
  x,
  PoR,
  type = c("none", "in", "out", "right", "left"),
  pb,
  plot = TRUE,
  ...
)
```

Arguments

`x`	`data.frame` or `sf` object containing the coordinates of the point(s) (`lat`, `lon`), the direction of σ_{Hmax} `azi` and its standard deviation `unc` (optional)
`PoR`	Pole of Rotation. `data.frame` or object of class `"euler.pole"` containing the geographical coordinates of the Euler pole
`type`	Character. Type of plate boundary (optional). Can be `"out"`, `"in"`, `"right"`, or `"left"` for outward, inward, right-lateral, or left-lateral moving plate boundaries, respectively. If `"none"` (the default), only the PoR-equivalent azimuth is returned.
`pb`	(optional) `sf` object of the plate boundary geometries in the geographical coordinate system
`plot`	(logical). Whether to produce a plot additional to output.
`...`	optional arguments to `distance_from_pb()`

Value

list containing the following values:

`results` data.frame showing the the coordinate and azimuth conversions (`lat.PoR`, `lon.PoR`, and `azi.PoR`), the predicted azimuths (`prd`), deviation angle from predicted (`dev`), circular distance (`cdist`), misfit to predicted stress direction (`nchisq`) and, if given, distance to tested plate boundary (`distance`)

`stats` array with circular (weighted) mean, circular standard deviation, circular variance, circular median, skewness, kurtosis, the 95% confidence angle, circular dispersion, and the normalized Chi-squared test statistic

`test` list containing the test results of the (weighted) Rayleigh test against the uniform distribution about the predicted orientation.

See Also

PoR_shmax(), distance_from_pb(), norm_chisq(), quick_plot(), circular_summary()

Examples

```
data("nuvel1")
na_pa <- subset(nuvel1, nuvel1$plate.rot == "na")

data("plates")
plate_boundary <- subset(plates, plates$pair == "na-pa")

data("san_andreas")
stress_analysis(san_andreas, na_pa, type = "right", plate_boundary, plot = TRUE)
```

---

stress_colors *Color palette for stress regime*

Description

Color palette for stress regime

Usage

```
stress_colors()
```

Value

function

Examples

```
stress_colors()
```

---

stress_data *Example crustal stress dataset*

Description

Subsets of the World Stress Map (WSM) compilation of information on the crustal present-day stress field (Version 1.1. 2019).

Usage

```
data('san_andreas')

data('tibet')

data('iceland')
```

Format

A `sf` object / `data.frame` with 10 columns. Each row represents a different in-situ stress measurement:

id Measurement identifier

lat Latitude in degrees

lon Longitude in degrees

azi SHmax azimuth in degrees

unc Measurement standard deviation (in degrees)

type Type of measurement

depth Depth in km

quality WSM quality rank

regime Stress regime

An object of class `sf` (inherits from `data.frame`) with 1126 rows and 10 columns.

An object of class `sf` (inherits from `data.frame`) with 1165 rows and 10 columns.

An object of class `sf` (inherits from `data.frame`) with 490 rows and 10 columns.

Details

`'san_andreas"` contains 407 stress data adjacent to the San Andreas Fault to be tested against a tangentially displaced plate boundary.

`"tibet"` contains 947 stress data from the Himalaya and Tibetan plateau to be tested against an inward-moving displaced plate boundary.

`'iceland` contains 201 stress data from Iceland to be tested against a outward-moving displaced plate boundary.

Source

https://www.world-stress-map.org/

References

Heidbach, O., Barth, A., Müller, B., Reinecker, J., Stephansson, O., Tingay, M., & Zang, A. (2016). WSM quality ranking scheme, database description and analysis guidelines for stress indicator. WSM Technical Report; 16-01. GFZ German Research Centre for Geosciences. doi:10.2312/WSM.2016.001

See Also

download_WSM() for description of columns and stress regime acronyms

Examples

```
data("san_andreas")
head(san_andreas)

data("tibet")
head(tibet)

data("iceland")
head(iceland)
```

---

stress_paths *Theoretical Plate Tectonic Stress Paths*

Description

Construct σ_{Hmax} lines that are following small circles, great circles, or loxodromes of an Euler pole for the relative plate motion.

Usage

```
eulerpole_paths(x, type = c("sc", "gc", "ld"), n = 10L, angle = 45, cw)

eulerpole_smallcircles(x, n = 10L)

eulerpole_greatcircles(x, n = 10L)

eulerpole_loxodromes(x, n = 10L, angle = 45, cw)
```

Arguments

x	Either an object of class "euler.pole" or "data.frame" containing coordinates of Euler pole in lat, lon, and rotation angle (optional).
type	character. String specifying the type of curves to export. Either "sm" for small circles (default), "gc" for great circles, or "ld" for loxodromes.
n	integer. Number of equally spaced curves; n = 10 by default (angular distance between curves: 180 / n)
angle	numeric. Direction of loxodromes; angle = 45 by default.
cw	logical. Sense of loxodromes: TRUE for clockwise loxodromes (left-lateral displaced plate boundaries). FALSE for counterclockwise loxodromes (right-lateral displaced plate boundaries).

Details

Maximum horizontal stress can be aligned to three types of curves related to relative plate motion:

Small circles Lines that have a constant distance to the Euler pole. If `x` contains `angle`, output additionally gives absolute velocity on small circle (°/Myr -> km/Myr).

Great circles Paths of the shortest distance between the Euler pole and its antipodal position.

Loxodromes Lines of constant bearing, i.e. curves cutting small circles at a constant angle.

Value

`sf` object

Author(s)

Tobias Stephan

Examples

```
data("nuvel1")
por <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to
# Pacific plate

eulerpole_smallcircles(por)
eulerpole_greatcircles(por)
eulerpole_loxodromes(x = por, angle = 45, n = 10, cw = FALSE)
eulerpole_loxodromes(x = por, angle = 30, cw = TRUE)
eulerpole_smallcircles(data.frame(lat = 30, lon = 10))
```

---

striae *Glacial striae from Finland*

Description

Directions of 51 glacial striations, example data from Davis (1986), pp. 316

Format

numeric vector of length 51

References

Davis, J.C. (1986) Statistics and Data Analysis in Geology. 2nd Edition, John Wiley & Sons, New York.

Examples

```
data("striae")
striae
```

superimposed_shmax *SHmax direction resulting from multiple plate boundaries*

Description

Calculates a σ_{Hmax} direction at given coordinates, sourced by multiple plate boundaries. This first-order approximation is the circular mean of the superimposed theoretical directions, weighted by the rotation rates of the underlying PoRs.

Usage

```
superimposed_shmax(df, PoRs, types, absolute = TRUE, PoR_weighting = NULL)
```

Arguments

df	data.frame containing the coordinates of the point(s) (lat, lon).
PoRs	multirow data.frame or "euler.pole" object that must contain lat, lon and angle
types	character vector with length equal to number of rows in PoRs. Type of plate boundary. Must be "out", "in", "right", or "left" for outward, inward, right-lateral, or left-lateral moving plate boundaries, respectively.
absolute	logical. Whether the resultant azimuth should be weighted using the absolute rotation at the points or the angular rotation of the PoRs.
PoR_weighting	(optional) numeric vector with length equal to number of rows in PoRs. Extra weightings for the used PoRs.

Value

two column vector. azi is the resultant azimuth in degrees / geographical CRS), R is the resultant length.

See Also

model_shmax()

superimposed_shmax_PB() for considering distances to plate boundaries

Examples

```
data(san_andreas)
data(nuvel1)
pors <- subset(nuvel1, plate.rot %in% c("eu", "na"))
res <- superimposed_shmax(san_andreas, pors, types = c("in", "right"), PoR_weighting = c(2, 1))
head(res)
```

`superimposed_shmax_PB` *SHmax direction resulting from multiple plate boundaries considering distance to plate boundaries*

Description

Calculates a σ_{Hmax} direction at given coordinates, sourced by multiple plate boundaries. This first-order approximation is the circular mean of the superimposed theoretical directions, weighted by the rotation rates of the underlying PoRs, the inverse distance to the plate boundaries, and the type of plate boundary.

Usage

```
superimposed_shmax_PB(
  x,
  pbs,
  model,
  rotation_weighting = TRUE,
 type_weights = c(divergent = 1, convergent = 3, transform_L = 2, transform_R = 2),
  idp = 1
)
```

Arguments

`x`	grid. An object of `sf`, `sfc` or 2-column matrix
`pbs`	plate boundaries. `sf` object
`model`	`data.frame` containing the Euler pole parameters. See `equivalent_rotation()` for details.
`rotation_weighting`	logical.
`type_weights`	named vector.
`idp`	numeric. Weighting power of inverse distance. The higher the number, the less impact far-distant boundaries have. When set to `0`, no weighting is applied.

Value

two-column matrix. `azi` is the resultant azimuth (in degrees), `R` is the resultant length.

See Also

`superimposed_shmax()`

Examples

```
na_grid <- sf::st_make_grid(san_andreas, what = "centers", cellsize = 1)
na_plate <- subset(plates, plateA == "na" | plateB == "na")
cpm <- cpm_models[["NNR-MORVEL56"]]

# make divergent to ridge-push:
na_plate <- transform(na_plate, type = ifelse(na_plate$pair == "eu-na", "convergent", type))

res <- superimposed_shmax_PB(na_grid, na_plate, model = cpm, idp = 2)
head(res)
```

---

tectonicr.colors *Colors for input variables*

Description

assigns colors to continuous or categorical values for plotting

Usage

```
tectonicr.colors(
  x,
  n = 10,
  pal = NULL,
  categorical = FALSE,
  na.value = "grey",
  ...
)
```

Arguments

x	values for color assignment
n	integer. number of colors for continuous colors (i.e. 'categorical = FALSE").
pal	either a named vector specifying the colors for categorical values, or a color function. If NULL, default colors are RColorBrewer::brewer.pal() (categorical = TRUE) and viridis::viridis() (categorical = FALSE).
categorical	logical.
na.value	color for NA values (categorical).
...	optional arguments passed to palette function

Value

named color vector

Examples

```
val1 <- c("N", "S", "T", "T", NA)
tectonicr.colors(val1, categorical = TRUE)
tectonicr.colors(val1, pal = stress_colors(), categorical = TRUE)

val2 <- runif(10)
tectonicr.colors(val2, n = 5)
```

`uniform_scores_test` *Uniform scores test*

Description

q-sample uniform-scores test tests the equality of given distributions, i.e., whether samples of directions share the mean direction. The test is the distribution-free test of Mardia, Watson & Wheeler on the uniform scores $2\pi r_i/n$ of the ranks in the pooled sample.

Usage

```
uniform_scores_test(samples, alpha = 0.05, axial = TRUE)
```

Arguments

`samples`	list of samples of orientations
`alpha`	Significance level of the test ranging between 0 and 1. For `dist = "vonmises"` it must be one of `0.1`, `0.05`, `0.01` or `0` (only the tabulated p-value range is returned).
`axial`	logical. Whether the data are axial, i.e. π-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`). In case of axial data, the angles will be doubled for the test.

Details

$$W = 2\sum_{i=1}^{q} R_i^2/n_i$$

with $2(q-1)$ degrees of freedom, $R^* = 2(n-1)R_1^2/(n_1 n_2)$ for two samples. Ties within a sample get consecutive ranks, ties between samples the average rank.

Value

A list with class `"htest"`

See Also

Other Two-Sample-Tests: `ar_test()`, `equal_kappa_test()`, `walraff_test()`, `watson_two_sample`, `watson_wheeler_test_perm()`, `watson_williams_test()`

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
uniform_scores_test(list(x1, x2))

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
uniform_scores_test(list(sa.por$azi.PoR, rvm(100, 135, 10)))
```

---

vcross *Vector cross product*

Description

Vector or cross product

Usage

```
vcross(x, y)
```

Arguments

x, y	numeric vectors of length 3

Value

numeric vector of length 3

Examples

```
vcross(c(1, 2, 3), c(4, 5, 6))
```

---

vm_qqplot *von Mises Quantile-Quantile Plot*

Description

Produces a Q-Q plot of the data against a specified von Mises distribution to graphically assess the goodness of fit of the model.

Usage

```
vm_qqplot(
  x,
  w = NULL,
  axial = TRUE,
  mean = NULL,
  kappa = NULL,
  xlab = "von Mises quantile function",
  ylab = "Empirical quantile function",
  main = "von Mises Q-Q Plot",
  col = "#B63679FF",
  add_line = TRUE,
  ...
)
```

Arguments

x	numeric. Angles in degrees
w	numeric. optional weightings for x to estimate mean and kappa.
axial	Logical. Whether data are uniaxial (axial=FALSE)
mean	numeric. Circular mean of the von Mises distribution. If NULL, it will be estimated from x.
kappa	numeric. Concentration parameter of the von Mises distribution. If NULL, it will be estimated from x.
xlab, ylab, main	plot labels.
col	color for the dots.
add_line	logical. Whether to connect the points by straight lines?
...	graphical parameters

Value

plot

See Also

vonmises()

Other circ-qqplot: circular_qqplot()

Examples

```
set.seed(20250411)

# von Mises distribution
x_vm <- rvm(100, mean = 0, kappa = 2)
vm_qqplot(x_vm, pch = 20)

x_wcauchy <- rwcauchy(100, mean = 0, rho = 0.5)
```

```
vm_qqplot(x_wcauchy, pch = 20)

# circular uniform data
x_cunif <- rcunif(100)
vm_qqplot(x_cunif, pch = 20)
```

---

vonmises *The von Mises Distribution*

Description

Density, probability distribution function, quantiles, and random generation for the circular normal distribution with mean μ and kappa κ.

Usage

```
rvm(n, mean, kappa)

dvm(theta, mean, kappa, axial = FALSE, log = FALSE)

pvm(theta, mean, kappa, from = NULL, tol = 1e-20)

qvm(p, mean, kappa, from = NULL, tol = .Machine$double.eps^(0.6), ...)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean	numeric. The mean vector in degrees.
kappa	numeric. Concentration parameter in the range $[0, Inf]$
theta	numeric. Angular value in degrees
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
log	logical. If TRUE, probabilities p are given as $\log(p)$.
from	if NULL is set to $\mu - \pi$. This is the value from which the pvm and qvm are evaluated. in degrees.
tol	numeric. The precision in evaluating the distribution function or the quantile.
p	numeric. Vector of probabilities with values in $[0, 1]$.
...	parameters passed to stats::integrate().

Value

dvm gives the density, pvm gives the probability of the von Mises distribution function, rvm generates random deviates (in degrees), and qvm provides quantiles (in degrees).

See Also

cunif, wnorm, and wcauchy

Examples

```
set.seed(1)
x <- rvm(5, mean = 90, kappa = 2)

dvm(x, mean = 90, kappa = 2)
dvm(x, mean = 90, kappa = 2, axial = TRUE)

pvm(x, mean = 90, kappa = 2)
qvm(c(.25, .5, .75), mean = 90, kappa = 2)
```

---

walraff_test *Wallraff Test of Angular Distances*

Description

Performs the Wallraff test of angular distances or angular dispersion around the mean.

Usage

```
walraff_test(samples, ref = NULL, axial = TRUE)
```

Arguments

samples	list of samples of orientations
ref	a vector of angles (in degrees) used as reference to compute the angular distances from, in each group. It should contain either one element (a common reference for all groups) or as many elements as there are groups, *in the same order* as the elements of the list samples. If ref is NULL (the default), the mean angle of each group will be used as reference. In this situation, the Wallraff test becomes a comparison of angular dispersion around the mean
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.

Details

The Wallraff test of angular distances between two or more groups is performed and the results are printed. The null hypothesis is that distances are equal across groups.

The test proceeds by computing the angular distances from a reference angle, in each group. The angular distance between two angles is the circular range. Then the distances are compared with a usual rank sum test (Kruskal-Wallis, stats::kruskal.test). When there are only two groups, the

Wilcoxon-Mann-Whitney test could be used but stats::wilcox.test without continuity correction for the p-value is equivalent to stats::kruskal.test so only stats::kruskal.test is used here.

The Wallraff test is most frequently used to compare angular dispersion around the mean, between samples. In this case, the reference angle is the mean angle of each sample. This is the default here, when no reference angles are provided.

Value

A list with class "htest" containing the following components:

statistic	the chi-squared statistic from kruskal.test.
parameter	the degrees of freedom for the chi-squared statistic.
p.value	the p-value for the test.
method	a character string containing the name of the test.
data.name	a character string giving the name(s) of the data.

References

Batschelet, E (1981). Circular Statistics in Biology. chap. 6.10, p. 124

Zar, J H (2010). Biostatistical analysis. sec. 27.7-8, p. 643

See Also

Other Two-Sample-Tests: ar_test(), equal_kappa_test(), uniform_scores_test(), watson_two_sample, watson_wheeler_test_perm(), watson_williams_test()

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
walraff_test(list(x1, x2))

x1 <- rvm(100, mean = 90, kappa = 20)
x2 <- rvm(100, mean = 90, kappa = 10)
x3 <- rvm(100, mean = 90, kappa = 1)
walraff_test(list(x1, x2, x3))

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
walraff_test(list(sa.por$azi.PoR, rvm(100, 135, 10)))
```

watson_test *Watson's U^2 Test for Goodness-of-Fit against known Distribution*

Description

A non-parametric statistical test used for circular data to determine whether a sample fits a specified theoretical distribution

Usage

```
watson_test(
  x,
  alpha = 0.05,
  dist = c("uniform", "vonmises"),
  axial = TRUE,
  quiet = FALSE
)
```

Arguments

x	numeric vector. Values in degrees
alpha	Significance level of the test ranging between 0 and 1. For dist = "vonmises" it must be one of 0.1, 0.05, 0.01 or 0 (only the tabulated p-value range is returned).
dist	Distribution to test for. The default, "uniform", is the circular uniform distribution. "vonmises" tests the von Mises distribution.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.
quiet	logical. Prints the test's decision.

Details

Hypotheses:

Null Hypothesis (H_0): The circular sample comes from a specified theoretical distribution (such as a uniform distribution or a specific von Mises distribution).

Alternative Hypothesis (H_1): The circular sample does not follow the specified theoretical distribution.

Interpretation:

To interpret the output of Watson's U^2 test, compare your calculated U^2 test statistic to the critical value from Watson's goodness-of-fit/homogeneity tables at your chosen significance level (α, commonly set to 0.05), or check the resulting p-value:

- If $U^2_{\text{calculated}} > U^2_{\text{critical}}$ (or p < α): Reject the null hypothesis (H_0). Conclude that the data significantly deviates from the theoretical distribution.
- If $U^2_{\text{calculated}} \leq U^2_{\text{critical}}$ (or $p \geq \alpha$): Fail to reject the null hypothesis (H_0). There is not enough evidence to claim the data deviates from the expected model.

Value

A list with class "htest" containing the following components:

statistic the value of the U2 statistic (modified for dist = "uniform")

parameter the number of data

p.value the p-value for the test (dist = "uniform" only)

p.range the range of the p-value from the tabulated critical values (dist = "vonmises" only)

estimate the MLE of the kappa value for the von Mises distribution

alternative a character string describing the alternative hypothesis.

method a character string indicating what type of Watson test was performed.

data.name a character string giving the name(s) of the data.

critical the critical value at the significance level

alpha the significance level

reject logical. whether to reject the null-hypothesis based on the significance level.

Note

Watson's test statistic is a rotation-invariant Cramer - von Mises test.

References

Mardia and Jupp (2000). Directional Statistics. John Wiley and Sons.

See Also

vonmises and cunif

Other One-Sample-Tests: hodges_ajne_test(), kuiper_test(), norm_chisq(), rao_spacing_test(), rayleigh-test, weighted-rayleigh-test

Examples

```
# Example data from Mardia and Jupp (1999), pp. 93
watson_test(homing, axial = FALSE, alpha = .05)

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
watson_test(sa.por$azi.PoR, alpha = .05)
watson_test(sa.por$azi.PoR, alpha = .05, dist = "vonmises")
```

watson_two_sample *Watson's Two-Sample Test of Homogeneity*

Description

Performs Watson's test for homogeneity on two samples of circular data. `watson_two_test_perm()` uses permutation to estimate p-values.

Usage

```
watson_two_test(x, y, alpha = 0.05, axial = TRUE, quiet = FALSE)

watson_two_test_perm(x, y, axial = TRUE, n_perm = 1000L, alpha = NULL)
```

Arguments

x, y	numeric vectors. Angles in degrees
alpha	Significance level of the test ranging between 0 and 1. For `dist = "vonmises"` it must be one of `0.1`, `0.05`, `0.01` or `0` (only the tabulated p-value range is returned).
axial	logical. Whether the data are axial, i.e. π-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`). In case of axial data, the angles will be doubled for the test.
quiet	logical. Prints the test's decision.
n_perm	integer. Number of permutations

Details

A two-sample Watson's U^2 test determines whether two independent groups of circular data (angles or directions) come from the same underlying distribution. Ties (e.g. azimuths rounded to whole degrees) are handled by evaluating the difference of the empirical distribution functions only at distinct values, weighted by the number of tied observations.

Hypotheses:

Null Hypothesis (H_0): The two samples come from the same circular distribution (the two groups of angles share a common distribution around the circle).

Alternative Hypothesis (H_1): The two samples come from different circular distributions (the groups are oriented or dispersed differently around the circle).

Interpretation:

The Test Statistic (U^2) measures the distance between the cumulative distribution functions of the two circular samples. A larger U^2 value means the two sets of angles look more different from each other.

The p-value of `watson_two_test_perm()` represents the probability of getting a U² value as large as (or larger than) your observed value purely by chance, assuming the null hypothesis is true. It is calculated by shuffling the group labels across your data many times to build a reference permutation distribution.

Making a Decision:

- Low p-value ($p \leq \alpha$, usually 0.05): Reject the null hypothesis. Conclude that the two groups have significantly different circular distributions.
- High p-value ($p > \alpha$): Fail to reject the null hypothesis. There is not enough evidence to say the two groups are distributed differently around the circle.

Value

A list with class `"htest"`

Note

Critical values and p-values of `watson_two_test()` are obtained using the asymptotic distribution of the test statistic. They should only be used for combined sample sizes in excess of 17. Tables are available for smaller sample sizes and can be found in Mardia (1972) for instance; alternatively use `watson_two_test_perm()`.

See Also

Other Two-Sample-Tests: `ar_test()`, `equal_kappa_test()`, `uniform_scores_test()`, `walraff_test()`, `watson_wheeler_test_perm()`, `watson_williams_test()`

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
watson_two_test(x1, x2, axial = FALSE)
watson_two_test_perm(x1, x2, axial = FALSE)

data1 <- rvm(n=20, mean = 0, kappa=3)
data2 <- rvm(n=20, mean = 90, kappa=2)
watson_two_test(data1, data2, axial = FALSE)
watson_two_test_perm(data1, data2, axial = FALSE)


# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
watson_two_test(sa.por$azi, rvm(100, 135, 10), alpha = 0.05)
watson_two_test_perm(sa.por$azi, rvm(100, 135, 10), alpha = 0.05)
```

`watson_wheeler_test_perm` *Watson-Wheeler Test of Homogeneity*

Description

A non-parametric statistical test used to determine whether two independent samples of circular data (angles, directions, or periodic times) come from the same underlying population distribution. The difference between the samples can be in either the mean direction or the dispersion. P-values are estimated using permutation.

Usage

```
watson_wheeler_test_perm(x, y, axial = TRUE, n_perm = 1000L, alpha = NULL)
```

Arguments

`x, y`	numeric vectors. Angles in degrees
`axial`	logical. Whether the data are axial, i.e. π-periodical (`TRUE`, the default) or directional, i.e. 2π-periodical (`FALSE`). In case of axial data, the angles will be doubled for the test.
`n_perm`	integer. Number of permutations
`alpha`	Significance level of the test ranging between 0 and 1. For `dist = "vonmises"` it must be one of `0.1`, `0.05`, `0.01` or `0` (only the tabulated p-value range is returned).

Details

The statistic is $R^* = 2(n-1)(C_1^2 + S_1^2)/(n_1 n_2)$ on the uniform scores $2\pi r_i/n$ of the ranks r_i in the pooled sample (Mardia & Jupp, 2000, eq. 8.3.4). Tied values get their average rank.

Hypotheses:

Null Hypothesis (H_0) The samples come from identical populations (meaning both the mean direction and the dispersion/variance are homogeneous across groups).

Alternative Hypothesis (H_1): The samples come from different population distributions, which can be due to a difference in the mean direction, a difference in variance/concentration, or both.

Interpretation:

Test Statistic: Higher values indicate larger discrepancies between the angular distributions of your groups.

- If the p-value is less than your significance level (commonly α = 0.05), you **reject** the null hypothesis. This means you have strong evidence that the groups differ significantly in their central direction or spread around the circle.
- If the p-value is greater than 0.05, you **fail to reject** the null hypothesis, meaning there is no statistically significant evidence of difference among the groups.

Value

A list with class "htest"

Note

Important Considerations & Limitations:

- **Sensitivity to both mean and variance**: Because it detects differences in either mean or variance, a significant result does not automatically mean the mean angles are different; it could be driven entirely by differences in concentration (variance).
- For more than two samples, or the asymptotic χ^2 version, see `uniform_scores_test()`.

See Also

Other Two-Sample-Tests: `ar_test()`, `equal_kappa_test()`, `uniform_scores_test()`, `walraff_test()`, `watson_two_sample`, `watson_williams_test()`

Examples

```
set.seed(20250411)
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
watson_wheeler_test_perm(x1, x2, axial = FALSE)

data1 <- rvm(n=20, mean = 0, kappa=3)
data2 <- rvm(n=20, mean = 90, kappa=2)
watson_wheeler_test_perm(data1, data2, axial = FALSE)

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
watson_wheeler_test_perm(sa.por$azi.PoR, rvm(100, 135, 10))
```

---

watson_williams_test *Watson-Williams Test of Homogeneity of Means*

Description

Performs the Watson-Williams test for homogeneity of means between several samples of circular data.

Usage

```
watson_williams_test(samples, alpha = 0.05, axial = TRUE)
```

Arguments

samples	list of samples of orientations
alpha	Significance level of the test ranging between 0 and 1. For dist = "vonmises" it must be one of 0.1, 0.05, 0.01 or 0 (only the tabulated p-value range is returned).
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.

Details

The Watson-Williams test for the homogeneity of means between two or more groups is performed and the results are printed. The null hypothesis is that means are equal across groups.

The assumptions are that: (1) the samples are drawn from populations with a von Mises distribution; (2) the parameter of concentration has the same value in all populations; (3) this parameter is sufficiently large (i.e. > 2 for two samples, > 1 for more samples). Assumptions 2 and 3 are checked and a warning is issued if they are not met.

Value

A list with class "htest"

See Also

Other Two-Sample-Tests: ar_test(), equal_kappa_test(), uniform_scores_test(), walraff_test(), watson_two_sample, watson_wheeler_test_perm()

Examples

```
x1 <- c(35, 45, 50, 55, 60, 70, 85, 95, 105, 120)
x2 <- c(75, 80, 90, 100, 110, 130, 135, 140, 150, 160, 165)
watson_williams_test(list(x1, x2), axial = FALSE)

data1 <- rvm(n=20, mean = 0, kappa=3)
data2 <- rvm(n=20, mean = 90, kappa=2)
watson_williams_test(list(data1, data2), axial = FALSE)

# San Andreas Fault Data:
data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
watson_williams_test(list(sa.por$azi.PoR, rvm(100, 135, 10)))
```

wcauchy *The Wrapped Cauchy Distribution*

Description

Density, probability distribution function, quantiles, and random generation for the circular wrapped Cauchy distribution with mean μ and rho ρ

Usage

```
rwcauchy(n, mean, rho)

dwcauchy(theta, mean, rho, axial = FALSE, log = FALSE)

pwcauchy(
  theta,
  mean,
  rho,
  axial = FALSE,
  from = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)

qwcauchy(
  p,
  mean,
  rho,
  axial = FALSE,
  from = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean	numeric. The mean vector in degrees.
rho	numeric. Concentration parameter in the range (0, 1)
theta	numeric. Angular value in degrees
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
log	logical. If TRUE, probabilities p are given as $\log(p)$.
from	if NULL is set to $\mu - \pi$. This is the value from which the pvm and qvm are evaluated. in degrees.

lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical. If TRUE, probabilities p are given as log(p).
p	numeric. Vector of probabilities with values in $[0, 1]$.

Value

dwcauchy gives the density, pwcauchy gives the probability of the wrapped Cauchy distribution function, rwcauchy generates random deviates (in degrees), and qrwcauchy provides quantiles (in degrees).

See Also

cunif, wnorm, and vonmises

Examples

```
set.seed(1)
x <- rwcauchy(5, mean = 90, rho = exp(-1))

dwcauchy(x, mean = 90, rho = exp(-1))
dwcauchy(x, mean = 90, rho = exp(-1), axial = TRUE)

pwcauchy(x, mean = 90, rho = exp(-1))
qwcauchy(c(.25, .5, .75), mean = 90, rho = exp(-1))
```

---

weighted-rayleigh-test

Weighted Goodness-of-fit Test for Circular Data

Description

Weighted version of the Rayleigh test (or V0-test) for uniformity against a distribution with a priori expected von Mises concentration. weighted_rayleigh_perm() estimates p-values by Monte Carlo simulation of uniform samples with the same weights.

Usage

```
weighted_rayleigh(
  x,
  mu = NULL,
  w = NULL,
  axial = TRUE,
  alpha = 0.05,
  quiet = FALSE
)

weighted_rayleigh_perm(x, mu = NULL, w = NULL, axial = TRUE, n_perm = 1000L)

v0_test(x, mu, w = NULL, axial = TRUE, n_perm = 1000L)
```

Arguments

x	numeric vector. Values in degrees
mu	(optional) The specified or known mean direction (in degrees) in alternative hypothesis.
w	numeric vector weights of length length(x). If NULL, the non-weighted Rayleigh test is performed.
axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE). In case of axial data, the angles will be doubled for the test.
alpha	Significance level of the test ranging between 0 and 1.
quiet	logical. Prints the test's decision.
n_perm	integer. Number of simulated uniform samples.

Details

The Null hypothesis is uniformity (randomness). The alternative is a unimodal distribution with the specified mean direction mu (weighted V-test), or with an unknown mean direction if mu is NULL (weighted Rayleigh test). The null hypothesis is rejected if p.value < alpha.

Under the null hypothesis the weighted mean cosine has variance $\sum w_i^2/(2(\sum w_i)^2)$, so the asymptotic tests use the effective sample size $n_{\text{eff}} = (\sum w_i)^2/\sum w_i^2$, which equals n for equal weights.

Value

a list with the components:

R **or** C weighted mean resultant length or, if mu is specified, the weighted mean cosine of the deviations from mu. Large values reject uniformity.

statistic Test statistic

p.value significance level of the test statistic

n_eff effective sample size

weighted_rayleigh_perm() returns a list of class "htest".

See Also

Other One-Sample-Tests: hodges_ajne_test(), kuiper_test(), norm_chisq(), rao_spacing_test(), rayleigh-test, watson_test()

Examples

```
# Load data
data("cpm_models")
data(san_andreas)
PoR <- equivalent_rotation(cpm_models[["NNR-MORVEL56"]], "na", "pa")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
data("iceland")
PoR.ice <- equivalent_rotation(cpm_models[["NNR-MORVEL56"]], "eu", "na")
```

```
ice.por <- PoR_shmax(iceland, PoR.ice, "out")
data("tibet")
PoR.tib <- equivalent_rotation(cpm_models[["NNR-MORVEL56"]], "eu", "in")
tibet.por <- PoR_shmax(tibet, PoR.tib, "in")

# GOF test:
weighted_rayleigh(tibet.por$azi.PoR, mu = 90, w = 1 / tibet$unc)
weighted_rayleigh(ice.por$azi.PoR, mu = 0, w = 1 / iceland$unc)
weighted_rayleigh(sa.por$azi.PoR, mu = 135, w = 1 / san_andreas$unc)

weighted_rayleigh_perm(tibet.por$azi.PoR, mu = 90, w = 1 / tibet$unc)
weighted_rayleigh_perm(ice.por$azi.PoR, mu = 0, w = 1 / iceland$unc)
weighted_rayleigh_perm(sa.por$azi.PoR, mu = 135, w = 1 / san_andreas$unc)
```

`weighting` *Weighting Factors*

Description

Helper function to transform uncertainty angles into weighting factors

Usage

```
weighting(
  x,
  method = c("linear-inverse", "inverse", "cosine", "none"),
  max.err = 90,
  norm = TRUE
)
```

Arguments

`x`	numeric. Uncertainty angle in degrees.
`method`	character. One of `"linear-inverse"` (the default), `"inverse"`, `"cosine"`, or `"none"` (no transformation).
`max.err`	numeric. The maximum expected error for x (90 by default).
`norm`	logical. Whether weights should be normalized so that the sum of the weights is 1. `TRUE` by default

Details

Linear inverse: $w = 1 - x/\sigma$, where σ is the maximum error expected for x (e.g. $90°$).

Inverse: $w = 1/x$

Cosine: $w = \cos x$

Value

numeric

Examples

```
x <- seq(0, 90, 1)

plot(x, weighting(x, "inverse"),
  col = 1, type = "l",
  xlab = "Uncertainty angle in degrees", ylab = "weight"
)
lines(x, weighting(x, "cosine"), col = 2)
lines(x, weighting(x, "linear-inverse"), col = 3)
legend("topright",
  col = 1:3, lty = 1,
  legend = c("inverse", "cosine", "linear-inverse")
)
```

---

wnorm *The Wrapped Normal Distribution*

Description

Density, probability distribution function, quantiles, and random generation for the circular normal distribution with mean μ and standard deviation σ.

Usage

```
rwnorm(n, mean = 0, sd = 1)

dwnorm(theta, mean = 0, sd = 1, axial = FALSE, log = FALSE)

pwnorm(theta, mean = 0, sd = 1, axial = FALSE, from = NULL, ...)

qwnorm(
  p,
  mean = 0,
  sd = 1,
  axial = FALSE,
  from = NULL,
  tol = .Machine$double.eps^(0.6),
  ...
)
```

Arguments

n	number of observations. If `length(n) > 1`, the length is taken to be the number required.
mean	numeric. The mean vector in degrees.
sd	numeric. standard deviation of the (unwrapped) normal distribution in degrees.
theta	numeric. Angular value in degrees

axial	logical. Whether the data are axial, i.e. π-periodical (TRUE, the default) or directional, i.e. 2π-periodical (FALSE).
log	logical. If TRUE, probabilities p are given as $\log(p)$.
from	if NULL is set to $\mu - \pi$. This is the value from which the pvm and qvm are evaluated. in degrees.
...	optional parameters passed to underlying circular functions `circular::pwrappednormal()` and `circular::qwrappednormal()`
p	numeric. Vector of probabilities with values in $[0, 1]$.
tol	numeric. The precision in evaluating the distribution function or the quantile.

Value

dwnorm gives the density, pwnorm gives the probability of the wrapped normal distribution function, rwnorm generates random deviates (in degrees), and qwnorm provides quantiles (in degrees).

See Also

cunif, wnorm, wcauchy, and vonmises

Examples

```
set.seed(1)
x <- rwnorm(5, mean = 90, sd = 5)

dwnorm(x, mean = 90, sd = 5, axial = FALSE)
dwnorm(x, mean = 90, sd = 5, axial = TRUE)

pwnorm(x, mean = 90, sd = 5)
qwnorm(c(.25, .5, .75), mean = 90, sd = 5)
```

Index

∗ **One-Sample-Tests**
  hodges_ajne_test, 46
  kuiper_test, 51
  norm_chisq, 55
  rao_spacing_test, 79
  rayleigh-test, 80
  watson_test, 116
  weighted-rayleigh-test, 124
∗ **Two-Sample-Tests**
  ar_test, 6
  equal_kappa_test, 37
  uniform_scores_test, 110
  walraff_test, 114
  watson_two_sample, 118
  watson_wheeler_test_perm, 120
  watson_williams_test, 121
∗ **circ-qqplot**
  circular_qqplot, 16
  vm_qqplot, 111
∗ **circular-plot**
  plot_density, 63
  plot_points, 65
  rose, 88
  rose_geom, 90
  rose_stats, 91
∗ **datasets**
  cpm_models, 26
  homing, 47
  import_WSM, 48
  nuvel1, 57
  nuvel1_plates, 57
  pb2002, 61
  plates, 62
  stress_data, 103
  striae, 106

abs_vel, 4
angle-conversion, 5
angle_vectors, 5
ar_test, 6
ar_test(), *38*, *110*, *115*, *119*, *121*, *122*
atan2_spec *(*spec_atan*)*, 96
atan2d_spec *(*spec_atan*)*, 96
axes, 7
axes(), *72*
axial_dispersion *(*ort-eigen*)*, 58
axial_strength *(*ort-eigen*)*, 58

base::getwd(), *49*
base::tempdir(), *49*
boot::boot(), *13*

cartesian_to_geographical *(*coordinates*)*, 24
cartesian_to_geographical(), *25*
cartesian_to_spherical *(*coordinates2*)*, *25*
cartesian_to_spherical(), *25*
circle_mean_diff, 8
circle_stats, 9
circular::pwrappednormal(), *128*
circular::qwrappednormal(), *128*
circular_density, 11
circular_density(), *14*, *64*
circular_dispersion *(*dispersion*)*, 31
circular_dispersion(), *14*, *34*, *79*
circular_dispersion_boot, 13
circular_dispersion_boot(), *34*
circular_distance *(*dispersion*)*, 31
circular_distance(), *69*
circular_IQR *(*circle_stats*)*, 9
circular_IQR(), *92*
circular_mean *(*circle_stats*)*, 9
circular_mean(), *19*, *20*, *32*, *79*, *82*, *92*, *101*
circular_mean_difference *(*circle_mean_diff*)*, 8
circular_mean_difference_alt *(*circle_mean_diff*)*, 8
circular_median *(*circle_stats*)*, 9
circular_median(), *92*, *101*

circular_mode, 14
circular_mode(), *20*
circular_plot, 15
circular_qqplot, 16
circular_qqplot(), *112*
circular_quantiles *(*circle_stats*)*, 9
circular_quantiles(), *20*
circular_range, 17
circular_sample_median *(*sample_median*)*, 94
circular_sample_median(), *92*
circular_sample_median_deviation *(*sample_median*)*, 94
circular_sample_median_deviation(), *92*
circular_sd *(*circle_stats*)*, 9
circular_sd(), *20*, *32*, *92*, *101*
circular_sd2 *(*dispersion*)*, 31
circular_sd2(), *32*
circular_sd_error, 18
circular_sd_error(), *22*
circular_summary, 19
circular_summary(), *34*, *100*, *101*, *103*
circular_var *(*circle_stats*)*, 9
circular_var(), *20*, *32*
compact-grid, 20
compact_grid *(*compact-grid*)*, 20
compact_grid(), *74*, *101*
compact_grid2 *(*compact-grid*)*, 20
confidence, 21
confidence_angle *(*confidence*)*, 21
confidence_angle(), *20*
confidence_interval *(*confidence*)*, 21
confidence_interval(), *92*
confidence_interval_fisher, 23
confidence_interval_fisher(), *79*, *92*
coordinate_mod, 26
coordinates, 24
coordinates2, 25
cpm_models, 26
cunif, 28, *114*, *117*, *124*, *128*
cunif(), *17*

data2PoR, 29
dcunif *(*cunif*)*, 28
deg2rad *(*angle-conversion*)*, 5
deviation_norm, 29
deviation_shmax, 30
deviation_shmax(), *55*, *69*
dispersion, 31
dist_greatcircle, 36
dist_greatcircle(), *101*
distance_binned_stats, 33
distance_from_pb, 34
distance_from_pb(), *79*, *102*, *103*
download_WSM *(*import_WSM*)*, 48
download_WSM(), *105*
download_WSM2016 *(*import_WSM*)*, 48
dplyr::dplyr_tidy_select(), *21*
dvm *(*vonmises*)*, 113
dvm(), *12*
dwcauchy *(*wcauchy*)*, 123
dwcauchy(), *12*
dwnorm *(*wnorm*)*, 127
dwnorm(), *12*

earth_radius, 37
earth_radius(), *4*
equal_kappa_test, 37
equal_kappa_test(), *7*, *110*, *115*, *119*, *121*, *122*
equivalent_rotation, 39
equivalent_rotation(), *108*
est.kappa *(*estimate-kappa*)*, 40
est.kappa(), *12*, *20*, *63*, *100*
estimate-kappa, 40
euler_pole, 41
euler_pole(), *83*
eulerpole_greatcircles *(*stress_paths*)*, 105
eulerpole_loxodromes *(*stress_paths*)*, 105
eulerpole_paths *(*stress_paths*)*, 105
eulerpole_smallcircles *(*stress_paths*)*, 105

geographical_to_cartesian *(*coordinates*)*, 24
geographical_to_cartesian(), *25*
geographical_to_PoR *(*por_transformation*)*, 75
geographical_to_spherical *(*coordinates*)*, 24
geom_azimuth, 42
geom_azimuth(), *44*, *45*
geom_azimuthpoint, 43
geom_azimuthpoint(), *43*
get_azimuth, 45
get_azimuth(), *46*
ggplot2::aes(), *42*, *44*

ggplot2::cut_number(), *33*
ggplot2::cut_width(), *33*
ggplot2::geom_point(), *44, 45*
ggplot2::geom_spoke(), *42–45*
ggplot2::ggplot(), *42*, *44*
ggplot2::layer(), *42*
graphics::arrows(), *8*
graphics::polygon(), *90, 91*
graphics::segments(), *91*
graphics::symbols(), *15*

haversine *(*spherical_angle*)*, 97
haversine(), *36*, *71*
hodges_ajne_test, 46
hodges_ajne_test(), *52*, *56*, *80*, *82*, *117*, *125*
homing, 47

iceland *(*stress_data*)*, 103
import_WSM, 48
is.euler, 50

kernel_dispersion(), *21*
kruskal.test, *115*
kuiper_test, 51
kuiper_test(), *47*, *56*, *80*, *82*, *117*, *125*

latitude_modulo *(*coordinate_mod*)*, 26
line_azimuth, 52
lines_azimuths *(*line_azimuth*)*, 52
load_WSM *(*import_WSM*)*, 48
load_WSM(), *49*
load_WSM2016 *(*import_WSM*)*, 48
longitude_modulo *(*coordinate_mod*)*, 26

mean_resultant_length, 53
mean_resultant_length(), *19*, *22*, *82*
min(), *21*
model_shmax, 54
model_shmax(), *31*, *69*, *76*, *107*

norm_chisq, 55
norm_chisq(), *47*, *52*, *69*, *79, 80*, *82*, *103*, *117*, *125*
nuvel1, 57
nuvel1_plates, 57

ort-eigen, 58
ortensor2d, 59
ortensor2d(), *59*
orthodrome *(*spherical_angle*)*, 97
orthodrome(), *36*, *71*, *76*
ot_eigen2d *(*ort-eigen*)*, 58
ot_eigen2d(), *60*

parse_wsm, 61
parse_wsm_quality *(*parse_wsm*)*, 61
pb2002, 61
pcunif *(*cunif*)*, 28
plates, 62
plot_density, 63
plot_density(), *12*, *66*, *90–92*
plot_points, 65
plot_points(), *64*, *90–92*
PoR2Geo_azimuth, 67
PoR_azi, 68
PoR_azimuth *(*PoR_azi*)*, 68
PoR_coordinates, 69
PoR_coordinates(), *70*
PoR_crs, 70
PoR_distance, 71
PoR_map, 72
PoR_shmax *(*PoR_azi*)*, 68
PoR_shmax(), *55*, *67*, *72*, *76*, *79*, *103*
PoR_stress2grid, 73
PoR_stress2grid(), *21*, *101*
PoR_stress2grid_stats *(*PoR_stress2grid*)*, 73
PoR_to_geographical *(*por_transformation*)*, 75
por_transformation, 75
PositionCenterSpoke, *42*, *44*
prd_err, 76
principal_direction *(*ort-eigen*)*, 58
projected_pb_strike, 77
pvm *(*vonmises*)*, 113
pwcauchy *(*wcauchy*)*, 123
pwnorm *(*wnorm*)*, 127

qcunif *(*cunif*)*, 28
quantise_wsm_quality *(*parse_wsm*)*, 61
quaternion *(*relative_rotation*)*, 83
quick_plot, 78
quick_plot(), *103*
qvm *(*vonmises*)*, 113
qwcauchy *(*wcauchy*)*, 123
qwnorm *(*wnorm*)*, 127

rad2deg *(*angle-conversion*)*, 5
rao_spacing_test, 79

rao_spacing_test(), *47*, *52*, *56*, *82*, *117*, *125*
rayleigh-test, 80
rayleigh_test *(*rayleigh-test*)*, 80
rayleigh_test_perm *(*rayleigh-test*)*, 80
rcunif *(*cunif*)*, 28
relative_rotation, 83
relative_rotation(), *39*
roll_circstats, 87
roll_confidence *(*rolling_test*)*, 84
roll_dispersion *(*rolling_test*)*, 84
roll_dispersion_CI *(*rolling_test*)*, 84
roll_dispersion_sde *(*rolling_test*)*, 84
roll_normchisq *(*rolling_test*)*, 84
roll_rayleigh *(*rolling_test*)*, 84
rolling_test, 84
rose, 88
rose(), *64*, *66*, *91, 92*
rose_fan *(*rose_geom*)*, 90
rose_geom, *64*, *66*, *90*, 90, *92*
rose_line *(*rose_geom*)*, 90
rose_stats, 91
rose_stats(), *64*, *66*, *90, 91*
rotation *(*relative_rotation*)*, 83
rvm *(*vonmises*)*, 113
rwcauchy *(*wcauchy*)*, 123
rwnorm *(*wnorm*)*, 127

sample_circular_dispersion *(*sample_dispersion*)*, 93
sample_circular_dispersion(), *23*
sample_circular_distance *(*sample_dispersion*)*, 93
sample_circular_distance(), *9*, *18*
sample_circular_variance *(*sample_dispersion*)*, 93
sample_dispersion, 93
sample_median, 94
san_andreas *(*stress_data*)*, 103
second_central_moment, 95
second_central_moment(), *20*
sf::st_crs(), *71*
sf::st_distance(), *96*
shortest_distance_to_line, 96
smoothr::densify(), *35*, *77*
spec_atan, 96
spherical_angle, 97
spherical_to_cartesian *(*coordinates2*)*, 25
spherical_to_cartesian(), *25*
spherical_to_geographical *(*coordinates2*)*, 25
stats::density(), *12*, *64*
stats::integrate(), *113*
stats::kruskal.test, *114, 115*
stats::wilcox.test, *115*
stress2grid, 98
stress2grid(), *21*, *74*, *100*
stress2grid_stats *(*stress2grid*)*, 98
stress2grid_stats(), *21*, *100, 101*
stress_analysis, 102
stress_colors, 103
stress_data, 103
stress_paths, 105
striae, 106
superimposed_shmax, 107
superimposed_shmax(), *108*
superimposed_shmax_PB, 108
superimposed_shmax_PB(), *107*

tectonicr.colors, 109
tectonicr.colors(), *72*
tibet *(*stress_data*)*, 103
title, *15*, *64*, *66*

uniform_scores_test, 110
uniform_scores_test(), *7*, *38*, *115*, *119*, *121, 122*

v0_test *(*weighted-rayleigh-test*)*, 124
vcross, 111
vincenty *(*spherical_angle*)*, 97
vincenty(), *36*, *71*
vm_qqplot, 111
vm_qqplot(), *17*, *79*
vonmises, *28*, 113, *117*, *124*, *128*
vonmises(), *40*, *112*

walraff_test, 114
walraff_test(), *7*, *38*, *110*, *119*, *121, 122*
watson_test, 116
watson_test(), *47*, *52*, *56*, *80*, *82*, *125*
watson_two_sample, *7*, *38*, *110*, *115*, 118, *121, 122*
watson_two_test *(*watson_two_sample*)*, 118
watson_two_test_perm *(*watson_two_sample*)*, 118
watson_wheeler_test_perm, 120
watson_wheeler_test_perm(), *7*, *38*, *110*, *115*, *119*, *122*

watson_williams_test, 121
watson_williams_test(), *7*, *38*, *110*, *115*, *119*, *121*
wcauchy, *28*, *114*, 123, *128*
weighted-rayleigh-test, 124
weighted_rayleigh (weighted-rayleigh-test), 124
weighted_rayleigh(), *79*
weighted_rayleigh_perm (weighted-rayleigh-test), 124
weighting, 126
wnorm, *28*, *114*, *124*, 127, *128*

zoo::na.fill(), *86, 87*
zoo::rollapply(), *86, 87*