

# Package ‘tectonic’

November 2, 2023

**Title** Analyzing the Orientation of Maximum Horizontal Stress

**Version** 0.2.95

**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](https://doi.org/10.1038/s41598-023-42433-2)> and Wdowinski (1998) <[doi:10.1016/S0079-1946\(98\)00091-3](https://doi.org/10.1016/S0079-1946(98)00091-3)>.

**License** GPL (>= 3)

**URL** <https://tobiste.github.io/tectonic/>

**BugReports** <https://github.com/tobiste/tectonic/issues>

**Depends** R (>= 4.1.0)

**Imports** boot, dplyr, ggplot2, methods, RColorBrewer, sf, smoothr, spatstat.explore, spatstat.geom, spatstat.utils, terra, tidyr, viridis, zoo

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

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.2.3

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2023-11-02 04:10:02 UTC

**R topics documented:**

abs_vel	3
angle-conversion	4
angle_vectors	5
axes	5
circle_stats	6
circular_dispersion_boot	8
circular_sd_error	9
compact_grid	10
confidence	11
conjugate_Q4	12
coordinates	13
coordinates2	14
coordinate_mod	15
cpm_models	15
deviation_norm	16
deviation_shmax	17
dispersion	18
dispersion_grid	19
distance_from_pb	21
distance_mod	22
dist_greatcircle	23
earth_radius	24
equivalent_rotation	24
est.kappa	25
euler_pole	26
euler_to_Q4	26
get_azimuth	27
get_distance	28
get_projected_pb_strike	28
get_relrot	29
iceland	29
is.euler	30
is.Q4	30
kuiper_test	31
line_azimuth	32
mean_resultant_length	32
model_shmax	33
normalize_Q4	35
norm_chisq	35
nuvel1	36
nuvel1_plates	37
pb2002	37
plates	38
PoR2Geo_azimuth	39
PoR_coordinates	40
PoR_crs	40

PoR_map . . . . .	41
PoR_shmax . . . . .	42
PoR_stress2grid . . . . .	43
por_transformation_df . . . . .	45
por_transformation_quat . . . . .	46
por_transformation_sf . . . . .	46
prd_err . . . . .	47
product_Q4 . . . . .	48
projected_pb_strike . . . . .	49
Q4_to_euler . . . . .	50
quantise_wsm_quality . . . . .	50
quick_plot . . . . .	51
raster_transformation . . . . .	52
rayleigh_test . . . . .	53
relative_rotation . . . . .	54
rolling_test . . . . .	55
rolling_test_dist . . . . .	58
roll_circstats . . . . .	60
rose . . . . .	62
rose_geom . . . . .	64
rose_stats . . . . .	64
rotation_Q4 . . . . .	66
san_andreas . . . . .	66
spec_atan . . . . .	67
spherical_angle . . . . .	68
stress2grid . . . . .	69
stress_analysis . . . . .	71
stress_colors . . . . .	72
stress_paths . . . . .	73
superimposed_shmax . . . . .	74
tectonicr.colors . . . . .	75
tibet . . . . .	76
vcross . . . . .	76
vonmises . . . . .	77
watson_test . . . . .	78
weighted_rayleigh . . . . .	79

**Index****81**

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	<i>Angle Between Two Vectors</i>
---------------	----------------------------------

---

**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)
```

---

axes	<i>Plot axes</i>
------	------------------

---

**Description**

Plot axes

**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 <a href="#">graphics::arrows()</a> 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 <a href="#">graphics::arrows()</a>

**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\_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.
w	(optional) Weights. A vector of positive numbers and of the same length as x.
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or directional, i.e. $2\pi$ -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-median and quasi-quantiles if `length(x) %% 2 != 1` and `length(x) %% 4 == 0`, respectively.

**References**

- Mardia, K.V. (1972). *Statistics of Directional Data: Probability and Mathematical Statistics*. London: Academic 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**, 3–15, doi:10.1016/j.tecto.2009.07.023

**Examples**

```
x <- rvm(10, 0, 100) %% 180
unc <- stats::runif(100, 0, 10)
circular_mean(x, 1 / unc)
circular_var(x, 1 / unc)
circular_sd(x, 1 / unc)
circular_median(x, 1 / unc)
circular_quantiles(x, 1 / unc)
circular_IQR(x, 1 / unc)

data("san_andreas")
circular_mean(san_andreas$azi)
circular_mean(san_andreas$azi, 1 / san_andreas$unc)
circular_median(san_andreas$azi)
circular_median(san_andreas$azi, 1 / san_andreas$unc)
circular_quantiles(san_andreas$azi)
circular_quantiles(san_andreas$azi, 1 / san_andreas$unc)
circular_var(san_andreas$azi)
circular_var(san_andreas$azi, 1 / san_andreas$unc)
```

```

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_mean(sa.por$azi.PoR, 1 / san_andreas$unc)
circular_median(sa.por$azi.PoR, 1 / san_andreas$unc)
circular_var(sa.por$azi.PoR, 1 / san_andreas$unc)
circular_quantiles(sa.por$azi.PoR, 1 / san_andreas$unc)

```

---

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	numeric values in degrees.
y	numeric. The angle(s) about which the angles x disperse (in degrees).
w, w.y	(optional) Weights for x and y, respectively. A vector of positive numbers and of the same length as x.
R	The number of bootstrap replicates. positive integer (1000 by default).
conf.level	Level of confidence: $(1 - \alpha\%)/100$ . (0.95 by default).
...	optional arguments passed to <code>boot::boot()</code>

### 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 = 1 / san_andreas$unc)
circular_dispersion_boot(sa.por$azi.PoR, y = 135, w = 1 / san_andreas$unc, R = 1000)
```

---

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. The approximated standard error of the mean direction is computed by the mean resultant length and the MLE concentration parameter  $\kappa$ .

**Usage**

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

**Arguments**

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

**Value**

Angle in degrees

**References**

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 = 1 / san_andreas$unc)
```

---

compact\_grid

*Compact smoothed stress field*


---

**Description**

Filter smoothed stress field to smallest wavelength (R) for each coordinate

**Usage**

```
compact_grid(x, type = c("stress2grid", "dispersion_grid"))
```

**Arguments**

x	output of <code>stress2grid()</code> or <code>PoR_stress2grid()</code>
type	character. Type of the grid x. Either "stress2grid" of "dispersion_grid"

**Value**

sf object

**Examples**

```
data("san_andreas")
res <- stress2grid(san_andreas)
compact_grid(res)
```

---

confidence                      *Confidence Interval around the Mean Direction of Circular Data*

---

### 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.
conf.level	Level of confidence: $(1 - \alpha\%)/100$ . (0.95 by default).
w	(optional) Weights. A vector of positive numbers and of the same length as x.
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or directional, i.e. $2\pi$ -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

- 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)

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 = 1 / san_andreas$unc)
confidence_interval(sa.por$azi.PoR, w = 1 / san_andreas$unc)
```

---

conjugate\_Q4

*Conjugation of a Quaternion*


---

**Description**

Inverse rotation given by conjugated quaternion

**Usage**

```
conjugate_Q4(q, normalize = FALSE)
```

**Arguments**

q	object of class "quaternion"
normalize	logical. Whether a quaternion normalization should be applied (TRUE) or not (FALSE, the default).

**Value**

object of class "quaternion"

---

`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 three-column 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)
p <- c(50, 10)
spherical_to_cartesian(p)
```

---

coordinate_mod	<i>Coordinate Correction</i>
----------------	------------------------------

---

**Description**

Corrects the longitudes or latitudes to value between -180.0 and 180.0 or -90 and 90 degree

**Usage**

```
longitude_modulo(x)
```

```
latitude_modulo(x)
```

**Arguments**

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

**Value**

numeric

**Examples**

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

---

cpm_models	<i>Global model of current plate motions</i>
------------	--

---

**Description**

Compilation of global models for current plate motions, including NNR-NUVEL1A (DeMets et al., 1990), NNR-MORVEL56 (Argus et al., 2011), REVEL (Sella et al., 2002), GSRM2.1 (Kreemer et al., 2014) HS-NUVEL1A (Gripp and Gordon, 2002), and PB2002 (Bird, 2003)

**Usage**

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

**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`

**model** Model for current global plate motion

**References**

- 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). doi: 10.1029/2011GC003751.
- 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), 321–361. 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), 3849–3889. doi: 10.1002/2014GC005407.
- 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")
```

---

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)
```



**Arguments**

x numeric vector containing angles in degrees

**Value**

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

**Author(s)**

Tobias Stephan

**Examples**

```
deviation_norm(91)
deviation_norm(c(-91, NA, 23497349))
```

---

deviation_shmax	<i>Deviation of Observed and Predicted Directions of Maximum Horizontal Stress</i>
-----------------	--

---

**Description**

Calculate the angular difference between the observed and modeled direction of maximum horizontal stresses ( $\sigma_{Hmax}$ ) 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  $\sigma_{Hmax}$ , i.e. the return object from `model_shmax()`

obs Numeric vector containing the observed azimuth of  $\sigma_{Hmax}$ , same length as prd

**Value**

An object of class `data.frame`

**dev.gc** Deviation of observed stress from modeled  $\sigma_{Hmax}$  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  $\sigma_{Hmax}$ .

## 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,
  norm = FALSE,
  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.  $2\pi$ -periodical (FALSE).

na.rm	logical. 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.
norm	logical. Whether the dispersion should be normalized by the maximum possible angular difference.

**Value**

circular\_distance returns a numeric vector of positive numbers, circular\_dispersion returns a positive number.

**Note**

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

**References**

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

**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$azi.PoR, y = 135, w = 1 / san_andreas$unc)
```

**Description**

Stress field and wavelength analysis using circular dispersion (or other statistical estimators for dispersion)

**Usage**

```
dispersion_grid(
  x,
  stat = c("dispersion", "nchisq", "rayleigh"),
  grid = NULL,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2.5,
  min_data = 3,
  threshold = 1,
  arte_thres = 200,
  dist_threshold = 0.1,
  R_range = seq(100, 2000, 100),
  ...
)
```

**Arguments**

<code>x</code>	sf object containing <b>azi</b> Azimuth in degree <b>unc</b> Uncertainties of azimuth in degree <b>prd</b> Predicted value for azimuth
<code>stat</code>	The measurement of dispersion to be calculated. Either "dispersion" (default), "nchisq", or "rayleigh" for circular dispersion, normalized Chi-squared test statistic, or Rayleigh test statistic.
<code>grid</code>	(optional) Point object of class sf.
<code>lon_range, lat_range</code>	(optional) numeric vector specifying the minimum and maximum longitudes and latitudes (are ignored if "grid" is specified).
<code>gridsize</code>	Numeric. Target spacing of the regular grid in decimal degree. Default is 2.5. (is ignored if "grid" is specified)
<code>min_data</code>	Integer. Minimum number of data per bin. Default is 3
<code>threshold</code>	Numeric. Threshold for stat value (default is 1)
<code>arte_thres</code>	Numeric. Maximum distance (in km) of the gridpoint to the next datapoint. Default is 200
<code>dist_threshold</code>	Numeric. Distance weight to prevent overweight of data nearby (0 to 1). Default is 0.1
<code>R_range</code>	Numeric value or vector specifying the search radius (in km). Default is seq(50, 1000, 50)
<code>...</code>	optional arguments to <code>dist_greatcircle()</code>

**Value**

sf object containing

**lon,lat** longitude and latitude in degree  
**stat** output of function defined in stat  
**R** Search radius in km  
**mdr** Mean distance of datapoints per search radius  
**N** Number of data points in search radius

### See Also

[circular\\_dispersion\(\)](#), [norm\\_chisq\(\)](#), [rayleigh\\_test\(\)](#)

### Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
san_andreas_por <- san_andreas
san_andreas_por$azi <- PoR_shmax(san_andreas, PoR, "right")$azi.PoR
san_andreas_por$prd <- 135
dispersion_grid(san_andreas_por)
```

---

distance_from_pb	<i>Distance from plate boundary</i>
------------------	-------------------------------------

---

### Description

Absolute distance of data points from the nearest plate boundary in degree

### Usage

```
distance_from_pb(x, PoR, pb, tangential = FALSE, km = 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).
km	Logical. Whether the distance is expressed in kilometers (TRUE) or in degrees (FALSE, the default).
...	optional arguments passed to <a href="#">smoothr::densify()</a>

### 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

**References**

Wdowinski, S. (1998). A theory of intraplate tectonics. *Journal of Geophysical Research: Solid Earth*, 103(3), 5037–5059. <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)
```

---

distance\_mod

*Normalize angular distance on a sphere distance*

---

**Description**

Helper function to express angular distance on the sphere in the range of 0 to 180 degrees

**Usage**

```
distance_mod(x)
```

**Arguments**

x                    numeric, angular distance (in degrees)

**Value**

numeric vector

---

dist_greatcircle	<i>Distance between points</i>
------------------	--------------------------------

---

### 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 <b>"euclidean"</b> Euclidean distance (not great circle distance!)

### Value

numeric vector with length equal to length(lat1)

### See Also

[orthodrome\(\)](#), [haversine\(\)](#), [vincenty\(\)](#)

### Examples

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

```

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

```

---

earth_radius	<i>Earth's radius in km</i>
--------------	-----------------------------

---

**Description**

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

**Usage**

```
earth_radius()
```

**Value**

numeric value

---

equivalent_rotation	<i>Equivalent rotation</i>
---------------------	----------------------------

---

**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")
```

---

 est.kappa

*Concentration parameter of von Mises distribution*

---

**Description**

Computes the maximum likelihood estimate of  $\kappa$ , the concentration parameter of a von Mises distribution, given a set of angular measurements.

**Usage**

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

**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.
...	optional paramters passed to <code>circular_mean()</code>

**Value**

numeric.

**Examples**

```
est.kappa(rvm(100, 90, 10), w = 1 / runif(100, 0, 10))
```

---

euler_pole	<i>Euler pole object</i>
------------	--------------------------

---

**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)
```

---

euler_to_Q4	<i>Quaternion from Euler angle-axis representation for rotations</i>
-------------	--

---

**Description**

Quaternion from Euler angle-axis representation for rotations

**Usage**

```
euler_to_Q4(x, normalize = FALSE)
```

**Arguments**

x	"euler.pole" object
normalize	logical. Whether a quaternion normalization should be applied (TRUE) or not (FALSE, the default).

**Value**

object of class "quaternion"

---

get_azimuth	<i>Azimuth Between two Points</i>
-------------	-----------------------------------

---

**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  $\psi_1, \lambda_1$  is the start point,  $\psi_2, \lambda_2$  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])
```

---

get_distance	<i>Helper function to Distance from plate boundary</i>
--------------	--

---

**Description**

Helper function to Distance from plate boundary

**Usage**

```
get_distance(lon, lat, pb.coords, tangential, km)
```

**Arguments**

lon, lat	numeric vectors
pb.coords	matrix
tangential, km	logical

**See Also**

[distance\\_from\\_pb\(\)](#)

---

get_projected_pb_strike	<i>Helper function to get Distance from plate boundary</i>
-------------------------	--

---

**Description**

Helper function to get Distance from plate boundary

**Usage**

```
get_projected_pb_strike(lon, lat, pb.coords, pb.bearing, tangential)
```

**Arguments**

lon, lat, pb.bearing	
	numeric vectors
pb.coords	matrix
tangential	logical

**See Also**

[projected\\_pb\\_strike\(\)](#)

---

`get_relrot`*Helper function to Equivalent rotation*

---

**Description**

Helper function to Equivalent rotation

**Usage**

```
get_relrot(plate.rot, lat, lon, angle, fixed, fixed.ep)
```

**Arguments**

<code>plate.rot, fixed</code>	character or numeric
<code>lat, lon, angle</code>	numeric
<code>fixed.ep</code>	data.frame

**See Also**

[equivalent\\_rotation\(\)](#)

---

`iceland`*Example stress data for outward-moving displaced plate boundary*

---

**Description**

Subset of the World Stress Map (WSM) compilation of information on the crustal present-day stress field (Version 2016). Subset contains stress data of Iceland.

**Usage**

```
data('iceland')
```

**Format**

An object of class "sf"

**Source**

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

## References

Heidbach, O., M. Rajabi, X. Cui, K. Fuchs, B. Müller, 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](https://doi.org/10.1016/j.tecto.2018.07.007).

## Examples

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

---

is.euler	<i>Check if object is euler.pole</i>
----------	--------------------------------------

---

## Description

Check if object is euler.pole

## Usage

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

## Arguments

x                    object of class "euler.pole"

## Value

logical

---

is.Q4	<i>Check if object is quaternion</i>
-------	--------------------------------------

---

## Description

Check if object is quaternion

## Usage

```
is.Q4(x)
```

## Arguments

x                    object of class "quaternion"

## Value

logical

---

`kuiper_test`*Kuiper Test of Circular Uniformity*

---

## Description

Kuiper test for circular random distribution.

## Usage

```
kuiper_test(x, alpha = 0, axial = TRUE)
```

## Arguments

<code>x</code>	numeric vector containing the circular data which are expressed in degrees
<code>alpha</code>	Significance level of the test. Valid levels are 0.01, 0.05, and 0.1. This argument may be omitted (NULL, the default), in which case, a range for the p-value will be returned.
<code>axial</code>	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or circular, i.e. $2\pi$ -periodical (FALSE).

## Details

Kuiper's test statistic is a rotation-invariant Kolmogorov-type test statistic. The critical values of a modified Kuiper's test statistic are used according to the tabulation given in Stephens (1970).

If `statistic > p.value`, the null hypothesis is rejected. If not, randomness (uniform distribution) cannot be excluded.

## Value

list containing the test statistic `statistic` and the significance level `p.value`.

## Examples

```
# Example data from Mardia and Jupp (2001), pp. 93
pidgeon_homing <- c(55, 60, 65, 95, 100, 110, 260, 275, 285, 295)
kuiper_test(pidgeon_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	<i>Extract azimuths of line segments</i>
--------------	--

---

**Description**

Extract azimuths of line segments

**Usage**

```
line_azimuth(x)
lines_azimuths(x)
```

**Arguments**

x sf object of type "LINESTRING" or "MULTILINESTRING"

**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. 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", package = "tectonic")
subset(plates, pair == "af-eu") |>
  smoothr::densify() |>
  line_azimuth()

line_azimuth(plates)
```

---

mean_resultant_length	<i>Mean Resultant Length</i>
-----------------------	------------------------------

---

**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	<i>Theoretical Direction of Maximum Horizontal Stress in the geographical reference system.</i>
-------------	---

---

**Description**

Models the direction of maximum horizontal stress  $\sigma_{Hmax}$  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**

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

**Value**

data.frame  
**gc** Azimuth of modeled  $\sigma_{Hmax}$  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  $\sigma_{Hmax}$ . `PoR_shmax()` to calculate the azimuth of  $\sigma_{Hmax}$  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 want to model the SHmax direction:
point <- data.frame(lat = 45, lon = 20)

model_shmax(point, euler)
```

---

normalize_Q4	<i>Quaternion normalization</i>
--------------	---------------------------------

---

**Description**

Quaternion normalization

**Usage**

normalize\_Q4(q)

**Arguments**

q                      quaternion

**Value**

object of class "quaternion"

---

norm_chisq	<i>Normalized Chi-Squared Test for Circular Data</i>
------------	--

---

**Description**

A quantitative comparison between the predicted and observed directions of  $\sigma_{Hmax}$  is obtained by the calculation of the average azimuth and by a normalized  $\chi^2$  test.

**Usage**

norm\_chisq(obs, prd, unc)

**Arguments**

obs	Numeric vector containing the observed azimuth of $\sigma_{Hmax}$ , same length as prd
prd	Numeric vector containing the modeled azimuths of $\sigma_{Hmax}$ , i.e. the return object from model_shmax()
unc	Uncertainty of observed $\sigma_{Hmax}$ , either a numeric vector or a number

**Details**

The normalized  $\chi^2$  test is

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

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

**Examples**

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na") # North America relative to
# Pacific plate
data(san_andreas)
point <- data.frame(lat = 45, lon = 20)
prd <- model_shmax(point, PoR)
norm_chisq(obs = c(50, 40, 42), prd$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	<i>Plate Boundaries on the Earth</i>
---------------	--------------------------------------

---

**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")
```

---

pb2002	<i>Global model of current plate motions</i>
--------	--

---

**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")
```

---

PoR2Geo_azimuth	<i>Azimuth conversion from PoR to geographical coordinate reference system</i>
-----------------	--

---

## Description

Conversion of PoR azimuths into geographical azimuths

## Usage

```
PoR2Geo_azimuth(x, PoR)
```

## Arguments

x	data.frame containing the PoR equivalent azimuths (azi.PoR), and either the geographical coordinates of the point(s) or the PoR-equivalent coordinates.
PoR	data.frame containing the geographical location of the Euler pole (lat, lon)

## 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](https://doi.org/10.1038/s41598023424332).

## See Also

[PoR\\_shmax\(\)](#)

## Examples

```
data("nuvel1")
# North America relative to Pacific plate:
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
data("san_andreas")
head(san_andreas$azi)
san_andreas$azi.PoR <- PoR_shmax(san_andreas, PoR)
res.geo <- PoR2Geo_azimuth(san_andreas, PoR)
head(res.geo)
```

---

PoR_coordinates	<i>PoR coordinates</i>
-----------------	------------------------

---

**Description**

Retrieve the PoR equivalent coordinates of an object

**Usage**

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

**Arguments**

x	sf or data.frame containing lat and lon coordinates (lat, lon)
PoR	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Euler pole

**Value**

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")
san_andreas.por_sf <- PoR_coordinates(san_andreas, por)
head(san_andreas.por_sf)
san_andreas.por_df <- PoR_coordinates(sf::st_drop_geometry(san_andreas), por)
head(san_andreas.por_df)
```

---

PoR_crs	<i>PoR coordinate reference system</i>
---------	--

---

**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\_map

*Plot data in PoR map*

---

**Description**

Plot data in PoR map

**Usage**

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

deviation      logical. Whether the data should be color-coded according to the deviation from the prediction, or according to the stress regime?

...              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", deviation = TRUE)
```

---

PoR\_shmax

*Direction of Maximum Horizontal Stress in PoR reference system*

---

### Description

Models the direction of maximum horizontal stress  $\sigma_{Hmax}$  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  $\sigma_{Hmax}$ , the deviation, and the normalized  $\chi^2$  statistics.

### Usage

```
PoR_shmax(df, PoR, type = c("none", "in", "out", "right", "left"))
```

### Arguments

df              `data.frame` containing the coordinates of the point(s) (lat, lon), the direction of  $\sigma_{Hmax}$  azi and its standard deviation unc (optional)

PoR             "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.

## Details

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

## Value

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),

nchisq the Norm  $\chi^2$  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  $\sigma_{Hmax}$  in the geographical reference system. `deviation_shmax()` to compute the deviation of the modeled direction from the observed direction of  $\sigma_{Hmax}$ . `norm_chisq()` to calculate the normalized  $\chi^2$  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)
```

## 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,
  ...
)
```

**Arguments**

<code>x</code>	<code>sf</code> object containing <b>azi</b> SHmax in degree <b>unc</b> Uncertainties of SHmax in degree <b>type</b> Methods used for the determination of the orientation of SHmax
<code>PoR</code>	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Euler pole
<code>grid</code>	(optional) Point object of class <code>sf</code> .
<code>PoR_grid</code>	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.
<code>lon_range, lat_range</code>	(optional) numeric vector specifying the minimum and maximum longitudes and latitudes (are ignored if "grid" is specified).
<code>gridsize</code>	Numeric. Target spacing of the regular grid in decimal degree. Default is 2.5 (is ignored if "grid" is specified)
<code>...</code>	Arguments passed to <code>stress2grid()</code>

**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)
- azi** geographical mean SHmax in degree
- azi.PoR** PoR mean SHmax in degree
- sd** Standard deviation of SHmax 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)
```

---

por\_transformation\_df *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	"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.
PoR	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Euler pole

**Value**

"data.frame" with the transformed coordinates (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))
```

por\_transformation\_quat

*Conversion between PoR to geographical coordinate system using quaternions*

---

### Description

Helper function for the transformation from PoR to geographical coordinate system or vice versa

### Usage

```
geographical_to_PoR_quat(x, PoR)
```

```
PoR_to_geographical_quat(x, PoR)
```

### Arguments

x, PoR                    two-column vectors containing the lat and lon coordinates

### Value

two-element numeric vector

### Examples

```
ep.geo <- c(20, 33)
q.geo <- c(10, 45)
q.por <- geographical_to_PoR_quat(q.geo, ep.geo)
q.por
PoR_to_geographical_quat(q.por, ep.geo)
```

---

por\_transformation\_sf *Conversion between PoR to geographical coordinates of spatial data*

---

### Description

Transform spatial objects from PoR to geographical coordinate reference system and vice versa.

### Usage

```
PoR_to_geographical_sf(x, PoR)
```

```
geographical_to_PoR_sf(x, PoR)
```

**Arguments**

- x sf, SpatRast, or Raster\* object of the data points in geographical or PoR coordinate system
- PoR Pole of Rotation. "data.frame" or object of class "euler.pole" 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 translation factors.

**Value**

sf or SpatRast object of the data points in the transformed geographical or PoR coordinate system

**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_sf(san_andreas, PoR)
PoR_to_geographical_sf(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)
```

---

product\_Q4

*Product of quaternions*

---

**Description**

Helper function for multiplication of two quaternions. Concatenation of two rotations R1 followed by R2

**Usage**

```
product_Q4(q1, q2, normalize = FALSE)
```

**Arguments**

q1, q2	two objects of class "quaternion". first rotation R1 expressed by q1 followed by second rotation R2 expressed by q2
normalize	logical. Whether a quaternion normalization should be applied (TRUE) or not (FALSE, the default).

**Value**

object of class "quaternion"

**Note**

Multiplication is not commutative.



---

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 Euler pole
tangential	Logical. Whether the plate boundary is a tangential boundary (TRUE) or an inward and outward boundary (FALSE, the default).
...	optional arguments passed to <code>smoothr::densify()</code>

### Details

Useful to calculate the beta angle, i.e. the angle between SHmax 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
```

---

Q4\_to\_euler                    *Euler angle/axis from quaternion*

---

### Description

Euler angle/axis from quaternion

### Usage

```
Q4_to_euler(q)
```

### Arguments

q                    object of class "quaternion"

### Value

"euler.pole" object

---

quantise\_wsm\_quality    *Numerical values to World Stress Map Quality Ranking*

---

### Description

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

### Usage

```
quantise_wsm_quality(regime)
```

### Arguments

regime                    Either a string or a character vector of WSM quality ranking

### Value

"integer" or vector of type "integer"

### 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. *World Stress Map Technical Report 16-01*, GFZ German Research Centre for Geosciences. [doi:10.2312/wsm.2016.001](https://doi.org/10.2312/wsm.2016.001)

**Examples**

```
quantise_wsm_quality(c("A", "B", "C", "D", NA))
data("san_andreas")
quantise_wsm_quality(san_andreas$quality)
```

---

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, and a rose diagram of the frequency distribution of the azimuths.

**Usage**

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

**Arguments**

azi	numeric. Azimuth of $\sigma_{Hmax}$
distance	numeric. Distance to plate boundary
prd	numeric. the predicted direction of $\sigma_{Hmax}$
unc	numeric. Uncertainty of observed $\sigma_{Hmax}$ , 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  $\chi^2$  statistics as a function of the distance to the plate boundary. The red line shows the rolling  $\chi^2$  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\\_angle\(\)](#), [norm\\_chisq\(\)](#), [weighted\\_rayleigh\(\)](#)

**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$azi.PoR, d, res$prd, san_andreas$unc, san_andreas$regime)
```

---

raster\_transformation *Conversion between PoR to geographical coordinate reference system of raster data*

---

**Description**

Helper function to transform raster data set from PoR to geographical coordinates

**Usage**

```
geographical_to_PoR_raster(x, PoR)
```

```
PoR_to_geographical_raster(x, PoR)
```

**Arguments**

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

**Value**

terra "SpatRaster" object

---

rayleigh_test	<i>Rayleigh Test of Circular Uniformity</i>
---------------	---

---

### Description

Performs a Rayleigh test of uniformity (or randomness), assessing the significance of the mean resultant length. The alternative hypothesis is an unimodal distribution with unknown mean direction and unknown mean resultant length if `mu` is `NULL`. If `mu` is specified the alternative hypothesis is a unimodal distribution with a specified mean direction and unknown mean resultant length.

### Usage

```
rayleigh_test(x, mu = NULL, axial = TRUE)
```

### Arguments

<code>x</code>	numeric vector. Values in degrees
<code>mu</code>	(optional) The specified or known mean direction (in degrees) in alternative hypothesis
<code>axial</code>	logical. Whether the data are axial, i.e. $\pi$ -periodical ( <code>TRUE</code> , the default) or directional, i.e. $2\pi$ -periodical ( <code>FALSE</code> ).

### Details

If `statistic > p.value`, the null hypothesis is rejected, i.e. the length of the mean resultant differs significantly from zero. If not, randomness (uniform distribution) cannot be excluded.

### Value

a list with the components:

`statistic` mean resultant length

`p.value` significance level of the test statistic

`p.value2` modified significance level (Cordeiro and Ferrari, 1991)

### Note

Although the Rayleigh test is consistent against (non-uniform) von Mises alternatives, it is not consistent against alternatives with  $p = 0$  (in particular, distributions with antipodal symmetry, i.e. 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()`).

**References**

- Mardia and Jupp (2000). Directional Statistics. John Wiley and Sons.
- Wilkie (1983): Rayleigh Test for Randomness of Circular Data. Appl. Statist. 32, No. 3, pp. 311-312
- 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\\_mean\(\)](#), [norm\\_chisq\(\)](#), [kuiper\\_test\(\)](#), [watson\\_test\(\)](#)

**Examples**

```
# Example data from Mardia and Jupp (2001), pp. 93
pidgeon_homing <- c(55, 60, 65, 95, 100, 110, 260, 275, 285, 295)
rayleigh_test(pidgeon_homing, axial = FALSE)

# 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
)
rayleigh_test(finland_stria, axial = FALSE)
rayleigh_test(finland_stria, mu = 105, axial = FALSE)

# Example data from Mardia and Jupp (2001), 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)

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

---

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)
relative_rotation(b, a)
```

---

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 $\sigma_{Hmax}$ , same length as prd
prd	Numeric vector containing the modeled azimuths of $\sigma_{Hmax}$ , i.e. the return object from <code>model_shmax()</code>
unc	Uncertainty of observed $\sigma_{Hmax}$ , either a numeric vector or a number
width	integer specifying the window width (in numbers of observations) which is aligned to the original sample according to the <code>align</code> argument. If <code>NULL</code> , an optimal width is estimated.
by.column	logical. If <code>TRUE</code> , FUN is applied to each column separately.
partial	logical or numeric. If <code>FALSE</code> then FUN is only applied when all indexes of the rolling window are within the observed time range. If <code>TRUE</code> (default), then the subset of indexes that are in range are passed to FUN. A numeric argument to <code>partial</code> 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 <code>fill</code> argument of <code>zoo::na.fill()</code> for details
...	optional arguments passed to <code>zoo::rollapply()</code>
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 ( <code>TRUE</code> , the default) or directional, i.e. $2\pi$ -periodical ( <code>FALSE</code> ).
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)
roll_rayleigh(dat.PoR$azi.PoR, prd = 135, unc = dat$unc)
roll_dispersion(dat.PoR$azi.PoR, y = 135, w = 1 / dat$unc)
roll_confidence(dat.PoR$azi.PoR, w = 1 / dat$unc)

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

---

 rolling\_test\_dist

*Apply Rolling Functions using Circular Statistics*


---

**Description**

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

**Usage**

```
distroll_circstats(
  x,
  distance,
  FUN,
  width = NULL,
  min_n = 2,
  align = c("right", "center", "left"),
  w = NULL,
  sort = TRUE,
  ...
)

distroll_confidence(
```

```

    x,
    distance,
    w = NULL,
    width = NULL,
    min_n = 2,
    align = c("right", "center", "left"),
    sort = TRUE,
    ...
)

distroll_dispersion(
  x,
  y,
  w = NULL,
  w.y = NULL,
  distance,
  width = NULL,
  min_n = 2,
  align = c("right", "center", "left"),
  sort = TRUE,
  ...
)

distroll_dispersion_sde(
  x,
  y,
  w = NULL,
  w.y = NULL,
  distance,
  width = NULL,
  min_n = 2,
  align = c("right", "center", "left"),
  sort = TRUE,
  ...
)

```

### Arguments

x, y	vectors of numeric values in degrees. length(y) is either 1 or length(x)
distance	numeric. the independent variable along the values in x are sorted, e.g. the plate boundary distances
FUN	the function to be applied
width	numeric. the range across distance on which FUN should be applied on x. If NULL, then width is a number that separates the distances in 10 equal groups.
min_n	integer. The minimum values that should be considered in FUN (2 by default), otherwise NA.
align	specifies whether the index of the result should be left- or right-aligned or centered (default) compared to the rolling window of observations. This argument

	is only used if width represents widths.
w	numeric. the weighting for x
sort	logical. Should the values be sorted after distance prior to applying the function (TRUE by default).
...	optional arguments to FUN
w.y	numeric. the weighting for y

### Value

two-column vectors of (sorted) x and the rolled statistics along distance.

### 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"))

distroll_circstats(dat$azi.PoR, distance = dat$distance, w = 1 / dat$unc, FUN = circular_mean)
distroll_confidence(dat$azi.PoR, distance = dat$distance, w = 1 / dat$unc)
distroll_dispersion(dat$azi.PoR, y = 135, distance = dat$distance, w = 1 / dat$unc)
distroll_dispersion_sde(dat$azi.PoR, y = 135, distance = dat$distance, w = 1 / dat$unc, R = 100)
```

---

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

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

### 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$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuve11, nuve11$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)

```

---

 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,
  clockwise = TRUE,
  muci = TRUE,
  round_binwidth = 0,
  mtext = "N",
  main = NULL,
  sub = NULL,
  at = seq(0, 360 - 45, 45),
  col = "grey",
  dots = FALSE,
  dot_pch = 1,
  dot_cex = 1,
  dot_col = "grey",
  ...
)

```

### Arguments

<code>x</code>	Data to be plotted. A numeric vector containing angles (in degrees).
<code>weights</code>	Optional vector of numeric weights associated with <code>x</code> .
<code>binwidth</code>	The width of the bins (in degrees).
<code>bins</code>	number of arcs to partition the circle width. Overridden by <code>binwidth</code> .
<code>axial</code>	Logical. Whether data are uniaxial ( <code>axial=FALSE</code> ) or biaxial ( <code>TRUE</code> , the default).

<code>equal_area</code>	Logical. Whether the radii of the bins are proportional to the frequencies ( <code>equal_area=FALSE</code> , i.e. equal-angle) or proportional to the square-root of the frequencies ( <code>equal_area=TRUE</code> , the default).
<code>clockwise</code>	Logical. Whether angles increase in the clockwise direction ( <code>clockwise=TRUE</code> , the default) or anti-clockwise, counter-clockwise direction ( <code>FALSE</code> ).
<code>muci</code>	logical. Whether the mean and its 95% CI are added to the plot or not.
<code>round_binwidth</code>	integer. Number of decimal places of bin width (0 by default).
<code>mtext</code>	character. String to be drawn at the top margin of the plot ("N" by default)
<code>main, sub</code>	Character string specifying the title and subtitle of the plot. If <code>sub = NULL</code> , it will show the bin width.
<code>at</code>	Optional vector of angles at which tick marks should be plotted. Set <code>at=numeric(0)</code> to suppress tick marks.
<code>col</code>	fill color of bins
<code>dots</code>	logical. Whether a circular dot plot should be added ( <code>FALSE</code> is the default).
<code>dot_cex, dot_pch, dot_col</code>	Plotting arguments for circular dot plot
<code>...</code>	Additional arguments passed to <code>spatstat.explore::rose()</code> .

**Value**

A window (class "owin") containing the plotted region.

**Note**

If `bins` and `binwidth` are `NULL`, an optimal bin width will be calculated using Scott (1979):

$$w_b = \frac{R}{n^{\frac{1}{3}}}$$

with `n` being the length of `x`, and the range `R` being either 180 or 360 degree for axial or directional data, respectively.

If `"axial" == TRUE`, the binwidth is adjusted to guarantee symmetrical fans.

**Examples**

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

data("san_andreas")
rose(san_andreas$azi, dots = TRUE, main = "dot plot")
rose(san_andreas$azi, weights = 1 / san_andreas$unc, main = "weighted")
```

---

rose_geom	<i>Lines and fans in rose diagram</i>
-----------	---------------------------------------

---

**Description**

Lines and fans in rose 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 rose diagram
axial	Logical. Whether x are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
add	logical. Add to existing plot?
...	optional arguments passed to <code>graphics::segments()</code> or <code>graphics::polygon()</code>
d	width of a fan (in degrees)

**Value**

No return value, called for side effects

**Examples**

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

---

rose_stats	<i>Show Average Direction and Spread in Rose Diagram</i>
------------	--

---

**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"),
  spread = c("CI", "sd", "IQR"),
  avg.col = "#85112AFF",
  avg.lty = 2,
  avg.lwd = 1.5,
  spread.col = ggplot2::alpha("#85112AFF", 0.2),
  spread.border = FALSE,
  spread.lty = NULL,
  spread.lwd = NULL,
  add = TRUE,
  ...
)

```

**Arguments**

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

**Value**

No return value, called for side effects

**See Also**

[rose\(\)](#) for plotting the rose diagram, and [circular\\_mean\(\)](#), [circular\\_median\(\)](#), [confidence\\_interval\(\)](#), [circular\\_sd\(\)](#), [circular\\_IQR\(\)](#) for statistical parameters.

**Examples**

```
data("san_andreas")
rose(san_andreas$azi, weights = 1 / san_andreas$unc, mucf = FALSE)
rose_stats(san_andreas$azi, weights = 1 / san_andreas$unc, avg = "median", spread = "IQR")
```

---

rotation_Q4	<i>Rotation of a vector by a quaternion</i>
-------------	---

---

**Description**

Rotation of a vector by a quaternion

**Usage**

```
rotation_Q4(q, p)
```

**Arguments**

q	object of class "quaternion"
p	three-column vector (Cartesian coordinates) of unit length

**Value**

three-column vector (Cartesian coordinates) of unit length

---

san_andreas	<i>Example stress data for tangentially displaced plate boundary</i>
-------------	--

---

**Description**

Subset of the World Stress Map (WSM) compilation of information on the crustal present-day stress field (Version 2016). Subset contains stress data adjacent to the San Andreas Fault.

**Usage**

```
data('san_andreas')
```

**Format**

An object of class "sf"

**Source**

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

## References

Heidbach, O., M. Rajabi, X. Cui, K. Fuchs, B. Müller, 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.

## Examples

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

---

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), 88–93. 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)
calgary <- c(51.04, -114.072)
orthodrome(berlin[1], berlin[2], calgary[1], calgary[2])
haversine(berlin[1], berlin[2], calgary[1], calgary[2])
vincenty(berlin[1], berlin[2], calgary[1], calgary[2])

```

---

stress2grid

*Spatial interpolation of SHmax*


---

**Description**

Stress field and wavelength analysis using (weighted) mean/median and standard deviation/IQR of stress data

**Usage**

```

stress2grid(
  x,
  stat = c("mean", "median"),
  grid = NULL,
  lon_range = NULL,
  lat_range = NULL,
  gridsize = 2.5,
  min_data = 3,
  threshold = 25,
  arte_thres = 200,
  method_weighting = FALSE,
  quality_weighting = TRUE,
  dist_weight = c("inverse", "linear", "none"),
  dist_threshold = 0.1,
  R_range = seq(50, 1000, 50),
  ...
)

```

**Arguments**

x	sf object containing <b>azi</b> SHmax in degree <b>unc</b> Uncertainties of SHmax in degree <b>type</b> Methods used for the determination of the direction of SHmax
stat	Should the direction of interpolated SHmax be based on the circular mean and standard deviation ("mean", the default) or on the circular median and interquartile range ("median")?
grid	(optional) Point object of class sf.

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)
min_data	Integer. Minimum number of data per bin. Default is 3
threshold	Numeric. Threshold for deviation of direction. Default is 25
arte_thres	Numeric. Maximum distance (in km) of the gridpoint to the next datapoint. Default is 200
method_weighting	Logical. If a method weighting should be applied: Default is FALSE.
quality_weighting	Logical. If a quality weighting should be applied: Default is TRUE.
dist_weight	Distance weighting method which should be used. One of "none", "linear", or "inverse" (the default).
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 search radius (in km). Default is seq(50, 1000, 50)
...	optional arguments to <code>dist_greatcircle()</code>

### Details

Updated version of the MATLAB script "stress2grid"

### Value

sf object containing

**lon,lat** longitude and latitude in degrees

**azi** Mean SHmax in degree

**sd** Standard deviation of SHmax in degrees

**R** Search radius in km

**mdr** Mean distance of datapoints per search radius

**N** Number of data points in search radius

### Source

<https://github.com/MorZieg/Stress2Grid>

### References

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

**See Also**

[dist\\_greatcircle\(\)](#), [PoR\\_stress2grid\(\)](#), [compact\\_grid\(\)](#), [circular\\_mean\(\)](#), [circular\\_median\(\)](#), [circular\\_sd\(\)](#)

**Examples**

```
data("san_andreas")
stress2grid(san_andreas)
```

---

stress_analysis	<i>Quick analysis of a stress data set</i>
-----------------	--

---

**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 $\sigma_{Hmax}$ 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 <a href="#">distance_from_pb()</a>

**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 dispersion, the 95% confidence angle, 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\(\)](#)

**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\_paths                      *Theoretical Plate Tectonic Stress Paths*

---

### Description

Construct  $\sigma_{Hmax}$  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 = 10, angle, cw)
```

```
eulerpole_smallcircles(x, n = 10)
```

```
eulerpole_greatcircles(x, n = 10)
```

```
eulerpole_loxodromes(x, n = 10, 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	Number of equally spaced curves; n = 10 by default (angular distance between curves: 180 / n)
angle	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 (degree/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))

```

---

superimposed\_shmax      *SHmax direction resulting from multiple plate boundaries*

---

**Description**

Calculates a  $\sigma_{Hmax}$  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), and the direction of $\sigma_{Hmax}$ azi (in degrees)
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**

numeric. Resultant azimuth in degrees and geographical CRS

**See Also**

[model\\_shmax\(\)](#)

**Examples**

```
data(san_andreas)
data(nuve11)
pors <- subset(nuve11, plate.rot %in% c("eu", "na"))
superimposed_shmax(san_andreas, pors, types = c("in", "right"), PoR_weighting = c(2, 1))
```

---

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 <code>RColorBrewer::brewer.pal()</code> (categorical = TRUE) and <code>viridis::viridis()</code> (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)
```

---

`tibet`*Example stress data for inward-moving displaced plate boundary*

---

**Description**

Subset of the World Stress Map (WSM) compilation of information on the crustal present-day stress field (Version 2016). Subset contains stress data of the Himalaya and Tibetan plateau

**Usage**

```
data('tibet')
```

**Format**

An object of class "sf"

**Source**

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

**References**

Heidbach, O., M. Rajabi, X. Cui, K. Fuchs, B. Müller, 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](https://doi.org/10.1016/j.tecto.2018.07.007).

**Examples**

```
data("tibet")
head(tibet)
```

---

`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))
```

---

vonmises

*The von Mises Distribution*

---

**Description**

Density, distribution function, and random generation for the circular normal distribution with mean and kappa.

**Usage**

```
rvm(n, mean, kappa)
```

```
dvm(theta, mean, kappa)
```

```
pvm(theta, mean, kappa)
```

**Arguments**

n	number of observations in degrees
mean	mean in degrees
kappa	concentration parameter
theta	angular value in degrees

**Value**

numeric vector.

**Examples**

```
x <- rvm(100, mean = 90, k = 100)
dvm(x, mean = 90, k = 100)
```

---

`watson_test`*Watson's  $U^2$  Test of Circular Uniformity*

---

### Description

Watson's test for circular random distribution.

### Usage

```
watson_test(  
  x,  
  alpha = 0,  
  dist = c("uniform", "vonmises"),  
  axial = TRUE,  
  mu = NULL  
)
```

### Arguments

<code>x</code>	numeric vector. Values in degrees
<code>alpha</code>	Significance level of the test. Valid levels are 0.01, 0.05, and 0.1. This argument may be omitted (NULL, the default), in which case, a range for the p-value will be returned.
<code>dist</code>	Distribution to test for. The default, "uniform", is the uniform distribution. "vonmises" tests the von Mises distribution.
<code>axial</code>	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or circular, i.e. $2\pi$ -periodical (FALSE).
<code>mu</code>	(optional) The specified or known mean direction (in degrees) in alternative hypothesis

### Details

Watson's test statistic is a rotation-invariant Cramer - von Mises test statistic. If `statistic > p.value`, the null hypothesis is rejected. If not, randomness (uniform distribution) cannot be excluded.

### Value

list containing the test statistic `statistic` and the significance level `p.value`.

### References

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

**Examples**

```
# Example data from Mardia and Jupp (2001), pp. 93
pidgeon_homing <- c(55, 60, 65, 95, 100, 110, 260, 275, 285, 295)
watson_test(pidgeon_homing, alpha = .05)

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

---

weighted_rayleigh	<i>Weighted Goodness-of-fit Test for Circular Data</i>
-------------------	--

---

**Description**

Weighted version of the Rayleigh test (or V0-test) for uniformity against a distribution with a priori expected von Mises concentration.

**Usage**

```
weighted_rayleigh(x, prd = NULL, unc, axial = TRUE)
```

**Arguments**

x	numeric vector. Values in degrees
prd	The a priori expected direction (in degrees) for the alternative hypothesis.
unc	numeric. The standard deviations of x. If NULL, the non-weighted Rayleigh test is performed.
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or directional, i.e. $2\pi$ -periodical (FALSE).

**Details**

The Null hypothesis is uniformity (randomness). The alternative is a distribution with a specified mean direction (prd). If `statistic > p.value`, the null hypothesis is rejected. If not, the alternative cannot be excluded.

**Value**

a list with the components:

`statistic` Test statistic

`p.value` significance level of the test statistic

**See Also**[rayleigh\\_test\(\)](#)**Examples**

```
# Load data
data("cpm_models")
data(san_andreas)
PoR <- equivalent_rotation(subset(cpm_models, model == "NNR-MORVEL56"), "na", "pa")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
data("iceland")
PoR.ice <- equivalent_rotation(subset(cpm_models, model == "NNR-MORVEL56"), "eu", "na")
ice.por <- PoR_shmax(iceland, PoR.ice, "out")
data("tibet")
PoR.tib <- equivalent_rotation(subset(cpm_models, model == "NNR-MORVEL56"), "eu", "in")
tibet.por <- PoR_shmax(tibet, PoR.tib, "in")

# GOF test:
weighted_rayleigh(tibet.por$azi.PoR, prd = 90, unc = tibet$unc)
weighted_rayleigh(ice.por$azi.PoR, prd = 0, unc = iceland$unc)
weighted_rayleigh(sa.por$azi.PoR, prd = 135, unc = san_andreas$unc)
```



# Index

## \* datasets

- cpm\_models, 15
  - iceland, 29
  - nuvel1, 36
  - nuvel1\_plates, 37
  - pb2002, 37
  - plates, 38
  - san\_andreas, 66
  - tibet, 76
- abs\_vel, 3
- angle-conversion, 4
- angle\_vectors, 5
- atan2\_spec (spec\_atan), 67
- atan2d\_spec (spec\_atan), 67
- axes, 5
- axes(), 42
- boot::boot(), 8
- cartesian\_to\_geographical  
(coordinates), 13
- cartesian\_to\_geographical(), 14
- cartesian\_to\_spherical (coordinates2),  
14
- cartesian\_to\_spherical(), 13
- circle\_stats, 6
- circular\_dispersion (dispersion), 18
- circular\_dispersion(), 9, 21, 52
- circular\_dispersion\_boot, 8
- circular\_distance (dispersion), 18
- circular\_distance(), 43
- circular\_IQR (circle\_stats), 6
- circular\_IQR(), 65
- circular\_mean (circle\_stats), 6
- circular\_mean(), 9, 19, 52, 54, 65, 71
- circular\_median (circle\_stats), 6
- circular\_median(), 65, 71
- circular\_quantiles (circle\_stats), 6
- circular\_sd (circle\_stats), 6
- circular\_sd(), 65, 71
- circular\_sd\_error, 9
- circular\_sd\_error(), 11
- circular\_var (circle\_stats), 6
- circular\_var(), 19
- compact\_grid, 10
- compact\_grid(), 45, 71
- confidence, 11
- confidence\_angle (confidence), 11
- confidence\_angle(), 52
- confidence\_interval (confidence), 11
- confidence\_interval(), 65
- conjugate\_Q4, 12
- coordinate\_mod, 15
- coordinates, 13
- coordinates2, 14
- cpm\_models, 15
- deg2rad (angle-conversion), 4
- deviation\_norm, 16
- deviation\_shmax, 17
- deviation\_shmax(), 34, 43
- dispersion, 18
- dispersion\_grid, 19
- dist\_greatcircle, 23
- dist\_greatcircle(), 20, 70, 71
- distance\_from\_pb, 21
- distance\_from\_pb(), 28, 52, 71, 72
- distance\_mod, 22
- distroll\_circstats (rolling\_test\_dist),  
58
- distroll\_confidence  
(rolling\_test\_dist), 58
- distroll\_dispersion  
(rolling\_test\_dist), 58
- distroll\_dispersion\_sde  
(rolling\_test\_dist), 58
- dvm (vonmises), 77
- earth\_radius, 24

- earth\_radius(), 4
- equivalent\_rotation, 24
- equivalent\_rotation(), 29
- est.kappa, 25
- euler\_pole, 26
- euler\_pole(), 55
- euler\_to\_Q4, 26
- eulerpole\_greatcircles (stress\_paths), 73
- eulerpole\_loxodromes (stress\_paths), 73
- eulerpole\_paths (stress\_paths), 73
- eulerpole\_smallcircles (stress\_paths), 73
  
- geographical\_to\_cartesian (coordinates), 13
- geographical\_to\_cartesian(), 14
- geographical\_to\_PoR (por\_transformation\_df), 45
- geographical\_to\_PoR\_quat (por\_transformation\_quat), 46
- geographical\_to\_PoR\_raster (raster\_transformation), 52
- geographical\_to\_PoR\_sf (por\_transformation\_sf), 46
- geographical\_to\_spherical (coordinates), 13
- get\_azimuth, 27
- get\_azimuth(), 27
- get\_distance, 28
- get\_projected\_pb\_strike, 28
- get\_relrot, 29
- graphics::arrows(), 6
- graphics::polygon(), 64
- graphics::segments(), 64
  
- haversine (spherical\_angle), 68
- haversine(), 23
  
- iceland, 29
- is.euler, 30
- is.Q4, 30
  
- kuiper\_test, 31
- kuiper\_test(), 53, 54
  
- latitude\_modulo (coordinate\_mod), 15
- line\_azimuth, 32
- lines\_azimuths (line\_azimuth), 32
  
- longitude\_modulo (coordinate\_mod), 15
  
- mean\_resultant\_length, 32
- mean\_resultant\_length(), 9, 11, 54
- model\_shmax, 33
- model\_shmax(), 18, 43, 48, 74
  
- norm\_chisq, 35
- norm\_chisq(), 21, 43, 52, 54, 72
- normalize\_Q4, 35
- nuvel1, 36
- nuvel1\_plates, 37
  
- orthodrome (spherical\_angle), 68
- orthodrome(), 23, 48
  
- pb2002, 37
- plates, 38
- PoR2Geo\_azimuth, 39
- PoR\_coordinates, 40
- PoR\_crs, 40
- PoR\_map, 41
- PoR\_shmax, 42
- PoR\_shmax(), 34, 39, 42, 48, 52, 72
- PoR\_stress2grid, 43
- PoR\_stress2grid(), 10, 71
- PoR\_to\_geographical (por\_transformation\_df), 45
- PoR\_to\_geographical\_quat (por\_transformation\_quat), 46
- PoR\_to\_geographical\_raster (raster\_transformation), 52
- PoR\_to\_geographical\_sf (por\_transformation\_sf), 46
- por\_transformation\_df, 45
- por\_transformation\_quat, 46
- por\_transformation\_sf, 46
- prd\_err, 47
- product\_Q4, 48
- projected\_pb\_strike, 49
- projected\_pb\_strike(), 28
- pvm (vonmises), 77
  
- Q4\_to\_euler, 50
- quantise\_wsm\_quality, 50
- quaternion (relative\_rotation), 54
- quick\_plot, 51
- quick\_plot(), 72
  
- rad2deg (angle-conversion), 4

raster\_transformation, 52  
rayleigh\_test, 53  
rayleigh\_test(), 21, 80  
relative\_rotation, 54  
relative\_rotation(), 25  
roll\_circstats, 60  
roll\_confidence (rolling\_test), 55  
roll\_dispersion (rolling\_test), 55  
roll\_dispersion\_CI (rolling\_test), 55  
roll\_dispersion\_sde (rolling\_test), 55  
roll\_normchisq (rolling\_test), 55  
roll\_rayleigh (rolling\_test), 55  
rolling\_test, 55  
rolling\_test\_dist, 58  
rose, 62  
rose(), 65  
rose\_fan (rose\_geom), 64  
rose\_geom, 64  
rose\_line (rose\_geom), 64  
rose\_stats, 64  
rotation (relative\_rotation), 54  
rotation\_Q4, 66  
rvm (vonmises), 77

san\_andreas, 66  
sf::st\_crs(), 41  
smoothr::densify(), 21, 49  
spatstat.explore::rose(), 63  
spec\_atan, 67  
spherical\_angle, 68  
spherical\_to\_cartesian (coordinates2),  
14  
spherical\_to\_cartesian(), 13  
spherical\_to\_geographical  
(coordinates2), 14  
stress2grid, 69  
stress2grid(), 10, 44, 45  
stress\_analysis, 71  
stress\_colors, 72  
stress\_paths, 73  
superimposed\_shmax, 74

tectonicr.colors, 75  
tectonicr.colors(), 42  
tibet, 76

vcross, 76  
vincenty (spherical\_angle), 68  
vincenty(), 23

vonmises, 77

watson\_test, 78  
watson\_test(), 53, 54  
weighted\_rayleigh, 79  
weighted\_rayleigh(), 52

zoo::na.fill(), 57, 61  
zoo::rollapply, 61  
zoo::rollapply(), 57