

# Package ‘tectonic’

August 21, 2024

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

**Version** 0.4.0

**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, circular (>= 0.5.0), dplyr, ggplot2, methods (>= 4.3.1), RColorBrewer, sf, smoothr (>= 1.0.1), spatstat.explore (>= 3.2.7), spatstat.geom (>= 3.2.9), spatstat.univar (>= 2.0.3), spatstat.utils (>= 3.0.4), terra, tidyr, viridis, zoo (>= 1.8.12)

**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** 7.3.1

**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** 2024-08-21 11:30:03 UTC

## Contents

abs_vel	4
angle-conversion	4
angle_vectors	5
axes	6
circle_mean_diff	7
circle_stats	8
circular_dispersion_boot	9
circular_qqplot	11
circular_range	12
circular_sd_error	12
circular_summary	14
compact_grid	15
confidence	15
confidence_interval_fisher	17
conjugate_Q4	18
coordinates	19
coordinates2	20
coordinate_mod	21
cpm_models	21
deviation_norm	22
deviation_shmax	23
dispersion	24
distance_from_pb	26
distance_mod	27
dist_greatcircle	28
earth_radius	29
equivalent_rotation	29
est.kappa	30
euler_pole	31
euler_to_Q4	31
get_azimuth	32
get_distance	33
get_projected_pb_strike	33
get_relrot	34
import_WSM2016	34
is.euler	36
is.Q4	36
kernel_dispersion	37
kuiper_test	38
line_azimuth	39
mean_resultant_length	40
mean_SC	41
model_shmax	42
normalize_Q4	43
norm_chisq	43
nuvell	44

nuvel1\_plates . . . . . 45

parse\_wsm . . . . . 46

pb2002 . . . . . 46

plates . . . . . 47

plot\_density . . . . . 48

plot\_points . . . . . 49

PoR2Geo\_azimuth . . . . . 50

PoR\_coordinates . . . . . 51

PoR\_crs . . . . . 52

PoR\_map . . . . . 53

PoR\_shmax . . . . . 54

PoR\_stress2grid . . . . . 55

por\_transformation\_df . . . . . 57

por\_transformation\_quat . . . . . 57

por\_transformation\_sf . . . . . 58

prd\_err . . . . . 59

product\_Q4 . . . . . 60

projected\_pb\_strike . . . . . 60

Q4\_to\_euler . . . . . 61

quick\_plot . . . . . 62

raster\_transformation . . . . . 63

rayleigh\_test . . . . . 63

relative\_rotation . . . . . 65

rolling\_test . . . . . 66

rolling\_test\_dist . . . . . 69

roll\_circstats . . . . . 71

rose . . . . . 73

rose\_geom . . . . . 75

rose\_stats . . . . . 76

rotation\_Q4 . . . . . 77

sample\_median . . . . . 78

second\_central\_moment . . . . . 79

spec\_atan . . . . . 80

spherical\_angle . . . . . 80

stress2grid . . . . . 81

stress\_analysis . . . . . 84

stress\_colors . . . . . 85

stress\_data . . . . . 85

stress\_paths . . . . . 87

superimposed\_shmax . . . . . 88

superimposed\_shmax\_PB . . . . . 89

tectonic.colors . . . . . 90

vcross . . . . . 91

vm\_qqplot . . . . . 92

vonmises . . . . . 93

watson\_test . . . . . 94

weighted\_rayleigh . . . . . 95

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

*Plot axes*

---

## Description

Plot axes

## Usage

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

## Arguments

<code>x, y</code>	coordinates of points
<code>angle</code>	Azimuth in degrees
<code>radius</code>	length of axis
<code>arrow.code</code>	integer. Kind of arrow head. The default is 1, i.e. no arrow head. See <a href="#">graphics::arrows()</a> for details
<code>arrow.length</code>	numeric Length of the edges of the arrow head (in inches). (Ignored if <code>arrow.code = 1</code> )
<code>add</code>	logical. add to existing plot?
<code>...</code>	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_mean_diff	<i>Circular Mean Difference</i>
------------------	---------------------------------

---

**Description**

Circular Mean Difference

**Usage**

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

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

**Arguments**

x	numeric vector. Values in degrees.
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

**References**

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

**Examples**

```
data("san_andreas")
circular_mean_difference(san_andreas$azi)
circular_mean_difference(san_andreas$azi, 1 / san_andreas$unc)

circular_mean_difference_alt(san_andreas$azi)
circular_mean_difference_alt(san_andreas$azi, 1 / san_andreas$unc)
```

**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)
```

```
sample_circular_dispersion(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  $\text{length}(x) \% 2 \neq 1$  and  $\text{length}(x) \% 4 == 0$ , respectively.

## References

- Mardia, K.V. (1972). *Statistics of Directional Data: Probability and Mathematical Statistics*. London: Academic Press.
- Mardia, K.V., and Jupp, P.E (1999). *Directional Statistics*, Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi:10.1002/9780470316979
- Ziegler, M. O.; Heidbach O. (2019). Manual of the Matlab Script Stress2Grid v1.1. *WSM Technical Report 19-02*, GFZ German Research Centre for Geosciences. doi:10.2312/wsm.2019.002
- Heidbach, O., Tingay, M., Barth, A., Reinecker, J., Kurfess, D., & Mueller, B. (2010). Global crustal stress pattern based on the World Stress Map database release 2008. *Tectonophysics* **482**, 3<U+2013>15, doi:10.1016/j.tecto.2009.07.023

## Examples

```
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_qqplot	<i>Quantile-Quantile Linearised Plot for Circular Distributions</i>
-----------------	---

---

### Description

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

### Usage

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

### Arguments

x	numeric. Angles in degrees
axial	Logical. Whether data are uniaxial (axial=FALSE)
xlab, ylab, main	plot labels.
...	graphical parameters

### Value

plot

### References

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

### Examples

```
x_vm <- rvm(100, mean = 0, kappa = 2)  
circular_qqplot(x_vm, pch = 20)  
  
x_norm <- rnorm(100, mean = 0, sd = 25)  
circular_qqplot(x_norm, pch = 20)  
  
x_unif <- runif(100, 0, 360)  
circular_qqplot(x_unif, pch = 20)
```

---

circular_range	<i>Circular Range</i>
----------------	-----------------------

---

**Description**

Length of the smallest arc which contains all the observations.

**Usage**

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

**Arguments**

x	numeric vector. Values in degrees.
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. angle in degrees

**References**

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

**Examples**

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

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

---

circular_sd_error	<i>Standard Error of Mean Direction of Circular Data</i>
-------------------	--

---

**Description**

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

**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**

numeric

**References**

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

Davis (1986) *Statistics and data analysis in geology*. 2nd ed., John Wiley & Sons.

**See Also**

[mean\\_resultant\\_length\(\)](#), [circular\\_mean\(\)](#)

**Examples**

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

data(san_andreas)
data("nuvell1")
PoR <- subset(nuvell1, nuvell1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_sd_error(sa.por$azi.PoR, w = 1 / san_andreas$unc)
```

---

circular\_summary      *Circular Summary statistics*

---

### Description

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

### Usage

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

### 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

named vector

### See Also

[circular\\_mean\(\)](#), [circular\\_sd\(\)](#), [circular\\_var\(\)](#), [circular\\_quantiles\(\)](#), [confidence\\_angle\(\)](#), [second\\_central\\_moment\(\)](#)

### Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_summary(sa.por$azi.PoR)
circular_summary(sa.por$azi.PoR, w = 1 / san_andreas$unc)
```

---

compact_grid	<i>Compact smoothed stress field</i>
--------------	--------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x	output of <a href="#">stress2grid()</a> , <a href="#">PoR_stress2grid()</a> , or <a href="#">kernel_dispersion()</a>
type	character. Type of the grid x. Either "stress" (when input is <a href="#">stress2grid()</a> or <a href="#">PoR_stress2grid()</a> ) or "dispersion" (when input is <a href="#">kernel_dispersion()</a> ).

**Value**

sf object

**See Also**

[stress2grid\(\)](#), [PoR\\_stress2grid\(\)](#), [kernel\\_dispersion\(\)](#)

**Examples**

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

---

confidence	<i>Confidence Interval around the Mean Direction of Circular Data</i>
------------	---

---

**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**

<code>x</code>	numeric vector. Values in degrees.
<code>conf.level</code>	Level of confidence: $(1 - \alpha\%)/100$ . (0.95 by default).
<code>w</code>	(optional) Weights. A vector of positive numbers and of the same length as <code>x</code> .
<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.

**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)
confidence_interval(finland_stria, axial = FALSE)

data(san_andreas)
data("nuvell1")
PoR <- subset(nuvell1, nuvell1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
confidence_angle(sa.por$azi.PoR, w = 1 / san_andreas$unc)
confidence_interval(sa.por$azi.PoR, w = 1 / san_andreas$unc)
```

---

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

---

### Description

For large samples ( $n \geq 25$ ) it performs a parametric estimate based on `sample_circular_dispersion()`. For smaller size samples, it returns a bootstrap estimate.

### Usage

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

### Arguments

<code>x</code>	numeric vector. Values in degrees.
<code>conf.level</code>	Level of confidence: $(1 - \alpha\%)/100$ . (0.95 by default).
<code>w</code>	(optional) Weights. A vector of positive numbers and of the same length as <code>x</code> .
<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>boot</code>	logical. Force bootstrap estimation
<code>R</code>	integer. number of bootstrap replicates
<code>quiet</code>	logical. Prints the used estimation (parametric or bootstrap).

### Value

list

### References

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

**Examples**

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

data(san_andreas)
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
confidence_interval_fisher(sa.por$azi.PoR, w = 1 / san_andreas$unc)
confidence_interval_fisher(sa.por$azi.PoR, w = 1 / san_andreas$unc, boot = TRUE)
```

---

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"

**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	<i>Coordinate Transformations</i>
--------------	-----------------------------------

---

**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, y = NULL)
```

**Arguments**

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

**Value**

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

**Author(s)**

Tobias Stephan

**Examples**

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

---

deviation_shmax	<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](https://doi.org/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
)

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

circular_dispersion_alt(
```

```

x,
y = NULL,
w = NULL,
w.y = NULL,
norm = FALSE,
axial = TRUE,
na.rm = TRUE
)

```

### Arguments

<code>x, y</code>	vectors of numeric values in degrees. <code>length(y)</code> is either 1 or <code>length(x)</code>
<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. Whether NA values in <code>x</code> should be stripped before the computation proceeds.
<code>w, w.y</code>	(optional) Weights. A vector of positive numbers and of the same length as <code>x</code> . <code>w.y</code> is the (optional) weight of <code>y</code> .
<code>norm</code>	logical. Whether the dispersion should be normalized by the maximum possible angular difference.

### Details

`circular_distance_alt()` and `circular_dispersion_alt()` are the alternative versions in Mardia and Jupp (2000), pp. 19-20. The alternative dispersion has a minimum at the sample median.

### Value

`circular_distance` returns a numeric vector of positive numbers, `circular_dispersion` returns a positive number.

### Note

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

### References

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

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

### See Also

`circular_mean()`, `circular_var()`.

**Examples**

```

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

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

data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
circular_dispersion(sa.por$azi.PoR, y = 135)
circular_dispersion(sa.por$azi.PoR, y = 135, w = 1 / san_andreas$unc)

```

---

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 <code>smoothr::densify()</code>

**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      *Distance between points*

---

### Description

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

### Usage

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

### Arguments

lat1, lon1	numeric vector. coordinate of point(s) 1 (degrees).
lat2, lon2	numeric vector. coordinates of point(s) 2 (degrees).
r	numeric. radius of the sphere (default = 6371.0087714 km, i.e. the radius of the Earth)
method	Character. Formula for calculating great circle distance, one of: <ul style="list-style-type: none"> <li>"haversine" great circle distance based on the haversine formula that is optimized for 64-bit floating-point numbers (the default)</li> <li>"orthodrome" great circle distance based on the spherical law of cosines</li> <li>"vincenty" distance based on the Vincenty formula for an ellipsoid with equal major and minor axes</li> <li><b>"euclidean"</b> Euclidean distance (not great circle distance!)</li> </ul>

### 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 parameters 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	<i>Helper function to Equivalent rotation</i>
------------	---

---

**Description**

Helper function to Equivalent rotation

**Usage**

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

**Arguments**

plate.rot, fixed	character or numeric
lat, lon, angle	numeric
fixed.ep	data.frame

**See Also**

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

---

import_WSM2016	<i>World Stress Map Database (WSM) Release 2016</i>
----------------	---

---

**Description**

Download WSM2016 database from the GFZ sever and applies optional filters. If destdir is specified, the data can be reloaded in a later R session using load\_WSM2016() using the same arguments.

**Usage**

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

```
load_WSM2016(
  file,
  quality = c("A", "B", "C", "D", "E"),
  lat_range = c(-90, 90),
  lon_range = c(-180, 180),
  depth_range = c(-Inf, Inf),
  method = c("B0", "BOC", "BOT", "BS", "DIF", "FMA", "FMF", "FMS", "GFI", "GFM", "GFS",
    "GVA", "HF", "HFG", "HFM", "HFP", "OC", "PC", "SWB", "SWL", "SWS"),
  regime = c("N", "NS", "T", "TS", "S", NA)
)
```

**Arguments**

destdir	where to save files, defaults to <code>base::tempdir()</code> , <code>base::getwd()</code> is also possible.
load	TRUE load the dataset into R, FALSE return the file name of the downloaded object.
...	(optional) arguments passed to <code>load_WSM2016()</code>
file	the name of the file which the data are to be read from.
quality	a character vectors containing the quality levels to be included. Includes all quality ranks (A-E) by default.
lat_range, lon_range	two-element numeric vectors giving the range of latitudes and longitudes (in degrees).
depth_range	two-element numeric vectors giving the depth interval (in km)
method	a character vectors containing the methods of stress inversion to be included. Includes all methods by default. See WSM2016 manual for used acronyms.
regime	a character vectors containing the stress regimes to be included. Acronyms: "N" - normal, "T" - thrust, "S" - strike-slip, "NS" - oblique normal, "TS" - oblique thrust, and NA - unknown faulting

**Value**

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

**Note**

Differences to the original dataset: All column names have been changed from uppercase in the original dataset to lowercase characters. Unknown azimuth values are represented by NA values instead of 999 in the original. Unknown regime is represented by NA instead of "U" in the original.

**Source**

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

**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**

```
## Not run:
download_WSM2016(
  quality = c("A", "B", "C"), lat_range = c(51, 72),
  lon_range = c(-180, -130), depth_range = c(0, 10), method = "FMS"
```

```
)  
## End(Not run)
```

---

is.euler

*Check if object is euler.pole*

---

**Description**

Check if object is euler.pole

**Usage**

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

**Arguments**

x                   object of class "euler.pole"

**Value**

logical

---

is.Q4

*Check if object is quaternion*

---

**Description**

Check if object is quaternion

**Usage**

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

**Arguments**

x                   object of class "quaternion"

**Value**

logical

---

kernel\_dispersion      *Adaptive Kernel Dispersion*

---

### Description

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

### Usage

```
kernel_dispersion(
  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

x	sf object containing <b>azi</b> Azimuth in degree <b>unc</b> Uncertainties of azimuth in degree <b>prd</b> Predicted value for azimuth
stat	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.
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 stat value (default is 1)
arte_thres	Numeric. Maximum distance (in km) of the grid point to the next data point. Default is 200

`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 (adaptive) kernel half-width(s) as search radius (in km). Default is `seq(50, 1000, 50)`

... optional arguments to `dist_greatcircle()`

**Value**

sf object containing

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

**stat** output of function defined in `stat`

**R** The 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
kernel_dispersion(san_andreas_por)
```

---

kuiper\_test

*Kuiper Test of Circular Uniformity*

---

**Description**

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).

**Usage**

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

**Arguments**

x	numeric vector containing the circular data which are expressed in degrees
alpha	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.
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or circular, i.e. $2\pi$ -periodical (FALSE).
quiet	logical. Prints the test's decision.

**Details**

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

## Not run:
lines_azimuths(plates)

## End(Not run)
```

---

mean\_resultant\_length *Mean Resultant Length*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

numeric.

**References**

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

**Examples**

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

---

mean_SC	<i>Mean cosines and sines</i>
---------	-------------------------------

---

**Description**

Mean cosines and sines

**Usage**

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

**Arguments**

x	angles in degrees
w	weightings
na.rm	logical

**Value**

named two element vector

**Examples**

```
## Not run:
x <- rvm(100, 0, 5)
mean_SC(x)

## End(Not run)
```

---

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

*Quaternion normalization***Description**

Quaternion normalization

**Usage**

```
normalize_Q4(q)
```

**Arguments**

q                      quaternion

**Value**

object of class "quaternion"

norm\_chisq

*Normalized Chi-Squared Test for Circular Data***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

*Plate Boundaries on the Earth*


---

**Description**

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

**Usage**

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

**Format**

An object of class `sf`

**References**

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

**Examples**

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

---

`parse_wsm`*Numerical values to World Stress Map Quality Ranking*

---

**Description**

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

**Usage**

```
parse_wsm_quality(x)
```

```
quantise_wsm_quality(x)
```

**Arguments**

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

**Value**

"numeric". the standard deviation of stress azimuth

**References**

Heidbach, O., Barth, A., 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**

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

---

`pb2002`*Global model of current plate motions*

---

**Description**

PB2002 global model of current plate motions by Bird 2003

**Usage**

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

**Format**

An object of class `data.frame`

**plate.name** The rotating plate

**plate.rot** The abbreviation of the plate's name

**lat,lon** Coordinates of the Pole of Rotation

**angle** The amount of rotation (angle in 1 Myr)

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

**source** Reference to underlying study

**References**

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

**Examples**

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

---

plates

*Plate Boundaries on the Earth*

---

**Description**

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

**Usage**

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

**Format**

An object of class `sf`

**References**

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

**Examples**

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

---

plot_density	<i>Circular density plot</i>
--------------	------------------------------

---

### Description

Plot the multiples of a von Mises density distribution

### Usage

```
plot_density(
  x,
  kappa,
  axial = TRUE,
  n = 512,
  norm_density = TRUE,
  ...,
  scale = 1.1,
  shrink = 1,
  add = TRUE,
  main = NULL,
  labels = TRUE,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE
)
```

### Arguments

x	Data to be plotted. A numeric vector containing angles (in degrees).
kappa	Concentration parameter for the von Mises distribution. Small kappa gives smooth density lines.
axial	Logical. Whether data are uniaxial ( <code>axial=FALSE</code> ) or biaxial ( <code>TRUE</code> , the default).
n	the number of equally spaced points at which the density is to be estimated.
norm_density	logical. Normalize the density?
...	Further graphical parameters may also be supplied as arguments.
scale	radius of plotted circle. Default is 1.1.
shrink	parameter that controls the size of the plotted function. Default is 1.
add	logical. Add to existing plot? ( <code>TRUE</code> by default).
main	Character string specifying the title of the plot.
labels	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
at	Optional vector of angles at which tick marks should be plotted. Set <code>at=numeric(0)</code> to suppress tick marks.
cborder	logical. Border of rose plot.

**Value**

plot or calculated densities as numeric vector

**See Also**

[dvm\(\)](#)

**Examples**

```
rose(san_andreas$azi, dots = TRUE, stack = TRUE, dot_cex = 0.5, dot_pch = 21)
plot_density(san_andreas$azi, kappa = 10, col = "seagreen", shrink = 1.5)
plot_density(san_andreas$azi, kappa = 10, col = "seagreen", add = FALSE, scale = .6)
```

---

plot\_points

*Add Points to a Circular Plot*

---

**Description**

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

**Usage**

```
plot_points(
  x,
  axial = TRUE,
  stack = FALSE,
  binwidth = 1,
  cex = 1,
  sep = 0.025,
  jitter_factor = 0,
  ...,
  scale = 1.1,
  add = TRUE,
  main = NULL,
  labels = TRUE,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE
)
```

**Arguments**

x	Data to be plotted. A numeric vector containing angles (in degrees).
axial	Logical. Whether data are uniaxial ( <code>axial=FALSE</code> ) or biaxial ( <code>TRUE</code> , the default).
stack	logical: if <code>TRUE</code> , points are stacked on the perimeter of the circle. Otherwise, all points are plotted on the perimeter of the circle. Default is <code>FALSE</code> .
binwidth	numeric. Bin width (in degrees) for the stacked dot plots. ignored when <code>stack==FALSE</code> . Is set to 1 degree by default.

<code>cex</code>	character (or symbol) expansion: a numerical vector. This works as a multiple of <code>par("cex")</code> .
<code>sep</code>	constant used to specify the distance between stacked points, if <code>stack==TRUE</code> or in the case of more than one dataset. Default is <code>0.025</code> ; smaller values will create smaller spaces.
<code>jitter_factor</code>	numeric. Adds a small amount of random variation to the location of each points along radius that is added to scale. Jitter is ignored when <code>stack==TRUE</code> ). If <code>0</code> , no jitter is added (by default); if negative, the points fall into the circle.
<code>...</code>	Further graphical parameters may also be supplied as arguments.
<code>scale</code>	radius of plotted circle. Default is <code>1.1</code> . Larger values shrink the circle, while smaller values enlarge the circle.
<code>add</code>	logical
<code>main</code>	Character string specifying the title of the plot.
<code>labels</code>	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
<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>cborder</code>	logical. Border of rose plot.

**Value**

A list with information on the plot

**Examples**

```
x <- rvm(100, mean = 90, k = 5)
plot_points(x, add = FALSE)
plot_points(x, jitter_factor = .2, add = FALSE) # jittered plot
plot_points(x, stack = TRUE, binwidth = 3, add = FALSE) # stacked plot
```

---

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**

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

**Value**

numeric vector of transformed azimuths (in degrees)

**References**

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

**See Also**

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

**Examples**

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

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

---

PoR\_coordinates

*PoR coordinates*

---

**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

*PoR coordinate reference system*


---

**Description**

Create the reference system transformed in Euler pole coordinate

**Usage**

```
PoR_crs(x)
```

**Arguments**

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

**Details**

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

**Value**

Object of class crs

**See Also**

[sf::st\\_crs\(\)](#)

**Examples**

```

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

```

---

PoR\_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"),
  show.deviation = FALSE,
  ...
)
```

**Arguments**

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

**Value**

plot

**See Also**

[PoR\\_shmax\(\)](#), [axes\(\)](#), [tectonicr.colors\(\)](#)

**Examples**

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

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

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

---

PoR\_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)
```

---

PoR\_stress2grid

*Spatial interpolation of SHmax in PoR coordinate reference system*

---

## Description

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

## Usage

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

## Arguments

**x** sf object containing  
**azi** SHmax in degree  
**unc** Uncertainties of SHmax in degree

	<b>type</b> Methods used for the determination of the orientation of SHmax
PoR	Pole of Rotation. "data.frame" or object of class "euler.pole" containing the geographical coordinates of the Euler pole
grid	(optional) Point object of class sf.
PoR_grid	logical. Whether the grid should be generated based on the coordinate range in the PoR (TRUE, the default) CRS or the geographical CRS (FALSE). Is ignored if grid is specified.
lon_range, lat_range	(optional) numeric vector specifying the minimum and maximum longitudes and latitudes (are ignored if "grid" is specified).
gridsize	Numeric. Target spacing of the regular grid in decimal degree. Default is 2.5 (is ignored if grid is specified)
...	Arguments passed to <a href="#">stress2grid()</a>

### 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)
```

---

pr<sub>d</sub>\_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 `smoothr::densify()`

**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

---

 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\\_interval\\_fisher\(\)](#), [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 *Rayleigh Test of Circular Uniformity*

---

**Description**

Performs a Rayleigh test for uniformity of circular/directional data by assessing the significance of the mean resultant length.

**Usage**

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

**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 (TRUE, the default) or directional, i.e. $2\pi$ -periodical (FALSE).
<code>quiet</code>	logical. Prints the test's decision.

**Details**

$H_0$ : angles are randomly distributed around the circle.

$H_1$ : angles are from unimodal distribution with unknown mean direction and mean resultant length (when `mu` is NULL. Alternatively (when `mu` is specified), angles are uniformly distributed around a specified direction.

If `statistic > p.value`, the null hypothesis is rejected, i.e. the length of the mean resultant differs significantly from zero, and the angles are not randomly distributed.

**Value**

a list with the components:

`R` or `C` mean resultant length or the dispersion (if `mu` is specified). Small values of `R` (large values of `C`) will reject uniformity. Negative values of `C` indicate that vectors point in opposite directions (also lead to rejection).

`statistic` test statistic

`p.value` significance level of the test statistic

**Note**

Although the Rayleigh test is consistent against (non-uniform) von Mises alternatives, it is not consistent against alternatives with  $p = 0$  (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\(\)](#), [weighted\\_rayleigh\(\)](#)

**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	<i>Relative rotation between two rotations</i>
-------------------	--

---

**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> , <code>FUN</code> is applied to each column separately.
partial	logical or numeric. If <code>FALSE</code> then <code>FUN</code> 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 <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.
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 <code>x</code> and <code>y</code> , respectively. A vector of positive numbers and of the same length as <code>x</code> .
conf.level	Level of confidence: $(1 - \alpha\%)/100$ . ( <code>0.95</code> 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

<code>x, y</code>	vectors of numeric values in degrees. <code>length(y)</code> is either 1 or <code>length(x)</code>
<code>distance</code>	numeric. the independent variable along the values in <code>x</code> are sorted, e.g. the plate boundary distances
<code>FUN</code>	the function to be applied
<code>width</code>	numeric. the range across <code>distance</code> on which <code>FUN</code> should be applied on <code>x</code> . If <code>NULL</code> , then <code>width</code> is a number that separates the distances in 10 equal groups.
<code>min_n</code>	integer. The minimum values that should be considered in <code>FUN</code> (2 by default), otherwise <code>NA</code> .
<code>align</code>	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 <code>width</code> represents widths.
<code>w</code>	numeric. the weighting for <code>x</code>
<code>sort</code>	logical. Should the values be sorted after <code>distance</code> 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$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuve11, nuve11$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**

x	numeric vector. Values in degrees.
w	(optional) Weights. A vector of positive numbers and of the same length as x.
FUN	the function to be applied
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or directional, i.e. $2\pi$ -periodical (FALSE).
na.rm	logical value indicating whether NA values in x should be stripped before the computation proceeds.
width	integer specifying the window width (in numbers of observations) which is aligned to the original sample according to the align argument. If NULL, an optimal width is calculated.
by.column	logical. If TRUE, FUN is applied to each column separately.
partial	logical or numeric. If FALSE then FUN is only applied when all indexes of the rolling window are within the observed time range. If TRUE (default), then the subset of indexes that are in range are passed to FUN. A numeric argument to partial can be used to determine the minimal window size for partial computations. See below for more details.
fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of <code>zoo::na.fill()</code> for details
...	optional arguments passed 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$pair == "na-pa")
data("san_andreas")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
distance <- distance_from_pb(
  x = san_andreas,
  PoR = PoR,
  pb = plate_boundary,
  tangential = TRUE
)
dat <- san_andreas[order(distance), ]
roll_circstats(dat$azi, w = 1 / dat$unc, circular_mean, width = 51)
```

---

rose	<i>Rose Diagram</i>
------	---------------------

---

**Description**

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

**Usage**

```
rose(
  x,
  weights = NULL,
  binwidth = NULL,
  bins = NULL,
  axial = TRUE,
  equal_area = TRUE,
  muci = TRUE,
  round_binwidth = 0,
  mtext = "N",
  main = NULL,
  sub = NULL,
  at = seq(0, 360 - 45, 45),
  cborder = TRUE,
  labels = TRUE,
  col = "grey",
  dots = FALSE,
  dot_pch = 1,
  dot_cex = 1,
  dot_col = "slategrey",
  stack = FALSE,
  jitter_factor = 0,
  add = FALSE,
  ...
)
```

**Arguments**

x	Data to be plotted. A numeric vector containing angles (in degrees).
weights	Optional vector of numeric weights associated with x.
binwidth	The width of the bins (in degrees).
bins	number of arcs to partition the circle width. Overridden by binwidth.
axial	Logical. Whether data are uniaxial ( <code>axial=FALSE</code> ) or biaxial ( <code>TRUE</code> , the default).
equal_area	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>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>cborder</code>	logical. Border of rose plot.
<code>labels</code>	Either a logical value indicating whether to plot labels next to the tick marks, or a vector of labels for the tick marks.
<code>col</code>	fill color of bins
<code>dots</code>	logical. Whether a circular dot plot should be added (FALSE is the default).
<code>dot_cex, dot_pch, dot_col</code>	Plotting arguments for circular dot plot
<code>stack</code>	logical. Groups and stacks the dots if TRUE. Default is FALSE.
<code>jitter_factor</code>	Add a small amount of noise to the angles' radius that is added to scale. Jitter is ignored when <code>stack==TRUE</code> ). If 0, no jitter is added (by default); if negative, the points fall into the circle.
<code>add</code>	logical.
<code>...</code>	Additional arguments passed to <code>spatstat.explore::rose()</code> .

**Value**

A window (class "owin") containing the plotted region or a list of the calculated frequencies.

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

# weighted frequencies:
```

```

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

# add dots
rose(san_andreas$azi, dots = TRUE, main = "dot plot", jitter = .2)
rose(san_andreas$azi,
     dots = TRUE, stack = TRUE, dot_cex = 0.5, dot_pch = 21,
     main = "stacked dot plot"
)

```

---

rose\_geom

*Direction Lines and Fans in Circular Diagram*


---

### Description

Direction Lines and Fans in Circular Diagram

### Usage

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

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

### Arguments

x	angles in degrees
radius	of the plotted circle
axial	Logical. Whether x are uniaxial (axial=FALSE) or biaxial (TRUE, the default).
add	logical. Add to existing plot?
...	optional arguments passed to <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

```

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

```

---

 rose\_stats

 Show Average Direction and Spread in Rose Diagram
 

---

### Description

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

### Usage

```
rose_stats(
  x,
  weights = NULL,
  axial = TRUE,
  avg = c("mean", "median", "sample_median"),
  spread = c("CI", "fisher", "sd", "IQR", "mdev"),
  avg.col = "#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), the circular Quasi Median (" <code>median</code> "), or the sample median (" <code>sample_median</code> ").
<code>spread</code>	character. The measure of spread to be plotted as a fan. Either 95% confidence interval (" <code>CI</code> ", the default), Fishers confidence interval (" <code>fisher</code> "), the circular standard deviation (" <code>sd</code> "), the Quasi interquartile range on the circle (" <code>IQR</code> "), or the sample median deviation (" <code>mdev</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

add                    logical.  
 ...                    optional arguments to `circular_plot()` if `add` is `FALSE`.

### Value

No return value, called for side effects

### See Also

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

### Examples

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

---

rotation\_Q4

*Rotation of a vector by a quaternion*

---

### 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

---

sample_median	<i>Circular Sample Median and deviation</i>
---------------	---

---

**Description**

Sample median direction for a vector of circular data

**Usage**

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

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

**Arguments**

x	numeric vector. Values in degrees.
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

**References**

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

**Examples**

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

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

---

second\_central\_moment *Second Central Momentum*

---

### Description

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

### Usage

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

### Arguments

x	numeric vector. Values in degrees.
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

Negative values of skewness indicate skewed data in counterclockwise direction.

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

### Value

list containing

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

std\_skewness standardized skewness

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

std\_kurtosis standardized kurtosis

### Examples

```
data("nuvel1")
PoR <- subset(nuvel1, nuvel1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
second_central_moment(sa.por$azi.PoR)
second_central_moment(sa.por$azi.PoR, w = 1 / san_andreas$unc)
```

---

spec_atan	<i>Quadrant-specific inverse of the tangent</i>
-----------	---

---

**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	<i>Angle along great circle on spherical surface</i>
-----------------	--

---

**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 interpolation and wavelength analysis using a kernel (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,
  min_data = 3L,
```

```

threshold = 25,
arte_thres = 200,
method_weighting = FALSE,
quality_weighting = TRUE,
dist_weight = c("inverse", "linear", "none"),
idp = 1,
qp = 1,
mp = 1,
dist_threshold = 0.1,
R_range = seq(50, 1000, 50),
...
)

```

### Arguments

<b>x</b>	sf object containing <b>azi</b> SHmax in degree <b>unc</b> (optional) Uncertainties of SHmax in degree <b>type</b> (optional) Methods used for the determination of the direction of SHmax
<b>stat</b>	whether the direction of interpolated SHmax is based on the circular mean and standard deviation ("mean", the default) or the circular median and interquartile range ("median")
<b>grid</b>	(optional) Point object of class sf.
<b>lon_range, lat_range</b>	(optional) numeric vector specifying the minimum and maximum longitudes and latitudes (ignored if <b>grid</b> is specified).
<b>gridsize</b>	numeric. Target spacing of the regular grid in decimal degree. Default is 2.5. (is ignored if <b>grid</b> is specified)
<b>min_data</b>	integer. Minimum number of data per bin. Default is 3
<b>threshold</b>	numeric. Threshold for deviation of direction. Default is 25
<b>arte_thres</b>	numeric. Maximum distance (in km) of the grid point to the next data point. Default is 200
<b>method_weighting</b>	logical. If a method weighting should be applied: Default is FALSE. If FALSE, overwrites <b>mp</b> .
<b>quality_weighting</b>	logical. If a quality weighting should be applied: Default is TRUE. If FALSE, overwrites <b>qp</b> .
<b>dist_weight</b>	Distance weighting method which should be used. One of "none", "linear", or "inverse" (the default).
<b>idp, qp, mp</b>	numeric. The weighting power of inverse distance, quality and method. Default is 1. The higher the value, the more weight it will put. When set to 0, no weighting is applied. <b>idp</b> is only effective if inverse distance weighting ( <b>dist_weight</b> ="inverse") is applied.
<b>dist_threshold</b>	numeric. Distance weight to prevent overweight of data nearby (0 to 1). Default is 0.1

**R\_range** numeric value or vector specifying the kernel half-width(s), i.e. the search radius (in km). Default is seq(50, 1000, 50)

... (optional) arguments to [dist\\_greatcircle\(\)](#)

### Details

This is a modified 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. 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, stat = "median")
```

---

stress\_analysis      *Quick analysis of a stress data set*

---

### Description

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

### Usage

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

### Arguments

x	data.frame or sf object containing the coordinates of the point(s) (lat, lon), the direction of $\sigma_{Hmax}$ 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 median, skewness, kurtosis, the 95% confidence angle, circular dispersion, and the normalized Chi-squared test statistic

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

**See Also**

[PoR\\_shmax\(\)](#), [distance\\_from\\_pb\(\)](#), [norm\\_chisq\(\)](#), [quick\\_plot\(\)](#), [circular\\_summary\(\)](#)

**Examples**

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

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

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

---

stress_colors	<i>Color palette for stress regime</i>
---------------	--

---

**Description**

Color palette for stress regime

**Usage**

```
stress_colors()
```

**Value**

function

**Examples**

```
stress_colors()
```

---

stress_data	<i>Example crustal stress dataset</i>
-------------	---------------------------------------

---

**Description**

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

**Usage**

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

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

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

**Format**

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

**id** Measurement identifier

**lat** latitude in degrees

**lon** longitude in degrees

**azi** SHmax azimuth in degrees

**unc** MMeasurement standard deviation (in degrees)

**type** Type of measurement

**depth** Depth in km

**quality** WSM quality rank

**regime** Stress regime

An object of class sf (inherits from data.frame) with 407 rows and 10 columns.

An object of class sf (inherits from data.frame) with 947 rows and 10 columns.

An object of class sf (inherits from data.frame) with 201 rows and 10 columns.

**Details**

'san\_andreas' contains 407 stress data adjacent to the San Andreas Fault to be tested against a tangentially displaced plate boundary.

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

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

**Source**

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

**References**

Heidbach, O., 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.

**See Also**

[download\\_WSM2016\(\)](#) for description of columns and stress regime acronyms

**Examples**

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

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

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

---

stress\_paths

*Theoretical Plate Tectonic Stress Paths*

---

**Description**

Construct  $\sigma_{Hmax}$  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 = 45, 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**

<code>df</code>	data.frame containing the coordinates of the point(s) ( <code>lat</code> , <code>lon</code> ), and the direction of $\sigma_{Hmax}$ <code>azi</code> (in degrees)
<code>PoRs</code>	multirow data.frame or "euler.pole" object that must contain <code>lat</code> , <code>lon</code> and <code>angle</code>

types	character vector with length equal to number of rows in PoRs. Type of plate boundary. Must be "out", "in", "right", or "left" for outward, inward, right-lateral, or left-lateral moving plate boundaries, respectively.
absolute	logical. Whether the resultant azimuth should be weighted using the absolute rotation at the points or the angular rotation of the PoRs.
PoR_weighting	(optional) numeric vector with length equal to number of rows in PoRs. Extra weightings for the used PoRs.

**Value**

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

**See Also**

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

[superimposed\\_shmax\\_PB\(\)](#) for considering distances to plate boundaries

**Examples**

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

---

superimposed\_shmax\_PB *SHmax direction resulting from multiple plate boundaries considering distance to plate boundaries*

---

**Description**

Calculates a  $\sigma_{Hmax}$  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, the inverse distance to the plate boundaries, and the type of plate boundary.

**Usage**

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

**Arguments**

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

**Value**

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

**See Also**

[superimposed\\_shmax\(\)](#)

**Examples**

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

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

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

---

tectonicr.colors      *Colors for input variables*

---

**Description**

assigns colors to continuous or categorical values for plotting

**Usage**

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

**Arguments**

<code>x</code>	values for color assignment
<code>n</code>	integer. number of colors for continuous colors (i.e. ‘categorical = FALSE’).
<code>pal</code>	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).
<code>categorical</code>	logical.
<code>na.value</code>	color for NA values (categorical).
<code>...</code>	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)
```

---

vcross

*Vector cross product*

---

**Description**

Vector or cross product

**Usage**

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

**Arguments**

<code>x, y</code>	numeric vectors of length 3
-------------------	-----------------------------

**Value**

numeric vector of length 3

**Examples**

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

---

 vm\_qqplot

*von Mises Quantile-Quantile Plot*


---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

plot

**Examples**

```
x_vm <- rvm(100, mean = 0, kappa = 4)
vm_qqplot(x_vm, axial = FALSE, pch = 20)

x_unif <- runif(100, 0, 360)
vm_qqplot(x_unif, axial = FALSE, pch = 20)
```

---

`vonmises`*The von Mises Distribution*

---

**Description**

Density, probability distribution function, quantiles, 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, from = NULL, tol = 1e-20)
```

```
qvm(p, mean = 0, kappa, from = NULL, tol = .Machine$double.eps^(0.6))
```

**Arguments**

<code>n</code>	number of observations in degrees
<code>mean</code>	mean in degrees
<code>kappa</code>	concentration parameter
<code>theta</code>	angular value in degrees
<code>from</code>	if NULL is set to $\mu - \pi$ . This is the value from which the pvm and qvm are evaluated. in degrees.
<code>tol</code>	the precision in evaluating the distribution function or the quantile.
<code>p</code>	numeric vector of probabilities with values in $[0, 1]$ .

**Value**

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

**Examples**

```
x <- rvm(100, mean = 90, kappa = 2)
dvm(x, mean = 90, kappa = 2)
pvm(x, mean = 90, kappa = 2)
qvm(c(.25, .5, .75), mean = 90, kappa = 2)
```

---

 watson\_test

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


---

### Description

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

### Usage

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

### Arguments

x	numeric vector. Values in degrees
alpha	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.
dist	Distribution to test for. The default, "uniform", is the uniform distribution. "vonmises" tests the von Mises distribution.
axial	logical. Whether the data are axial, i.e. $\pi$ -periodical (TRUE, the default) or circular, i.e. $2\pi$ -periodical (FALSE).
mu	(optional) The specified mean direction (in degrees) in alternative hypothesis
quiet	logical. Prints the test's decision.

### Details

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("nuvell1")
PoR <- subset(nuvell1, nuvell1$plate.rot == "na")
sa.por <- PoR_shmax(san_andreas, PoR, "right")
watson_test(sa.por$azi.PoR, alpha = .05)
watson_test(sa.por$azi.PoR, alpha = .05, dist = "vonmises")
```

---

weighted\_rayleigh

*Weighted Goodness-of-fit Test for Circular Data*


---

**Description**

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

**Usage**

```
weighted_rayleigh(x, mu = NULL, w = NULL, axial = TRUE, quiet = FALSE)
```

**Arguments**

x	numeric vector. Values in degrees
mu	The <i>a priori</i> expected direction (in degrees) for the alternative hypothesis.
w	numeric vector weights of length <code>length(x)</code> . 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).
quiet	logical. Prints the test's decision.

**Details**

The Null hypothesis is uniformity (randomness). The alternative is a distribution with a (specified) mean direction (`mu`). If `statistic >= p.value`, the null hypothesis of randomness is rejected and angles derive from a distribution with a (or the specified) mean direction.

**Value**

a list with the components:

**R** or **C** mean resultant length or the dispersion (if  $\mu$  is specified). Small values of R (large values of C) will reject uniformity. Negative values of C indicate that vectors point in opposite directions (also lead to rejection).

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$azi.PoR, mu = 90, w = 1 / tibet$unc)
weighted_rayleigh(ice.por$azi.PoR, mu = 0, w = 1 / iceland$unc)
weighted_rayleigh(sa.por$azi.PoR, mu = 135, w = 1 / san_andreas$unc)
```

# Index

## \* datasets

- cpm\_models, 21
  - import\_WSM2016, 34
  - nuvel1, 44
  - nuvel1\_plates, 45
  - pb2002, 46
  - plates, 47
  - stress\_data, 85
- abs\_vel, 4
- angle-conversion, 4
- angle\_vectors, 5
- atan2\_spec (spec\_atan), 80
- atan2d\_spec (spec\_atan), 80
- axes, 6
- axes(), 53
- base::getwd(), 35
- base::tempdir(), 35
- boot::boot(), 10
- cartesian\_to\_geographical  
(coordinates), 19
- cartesian\_to\_geographical(), 20
- cartesian\_to\_spherical (coordinates2),  
20
- cartesian\_to\_spherical(), 19
- circle\_mean\_diff, 7
- circle\_stats, 8
- circular\_dispersion (dispersion), 24
- circular\_dispersion(), 10, 38, 62
- circular\_dispersion\_alt (dispersion), 24
- circular\_dispersion\_alt(), 25
- circular\_dispersion\_boot, 9
- circular\_distance (dispersion), 24
- circular\_distance(), 55
- circular\_distance\_alt (dispersion), 24
- circular\_distance\_alt(), 25
- circular\_IQR (circle\_stats), 8
- circular\_IQR(), 77
- circular\_mean (circle\_stats), 8
- circular\_mean(), 13, 14, 25, 62, 64, 77, 83
- circular\_mean\_difference  
(circle\_mean\_diff), 7
- circular\_mean\_difference\_alt  
(circle\_mean\_diff), 7
- circular\_median (circle\_stats), 8
- circular\_median(), 77, 83
- circular\_qqplot, 11
- circular\_quantiles (circle\_stats), 8
- circular\_quantiles(), 14
- circular\_range, 12
- circular\_sample\_median (sample\_median),  
78
- circular\_sample\_median(), 77
- circular\_sample\_median\_deviation  
(sample\_median), 78
- circular\_sample\_median\_deviation(), 77
- circular\_sd (circle\_stats), 8
- circular\_sd(), 14, 77, 83
- circular\_sd\_error, 12
- circular\_sd\_error(), 16
- circular\_summary, 14
- circular\_summary(), 85
- circular\_var (circle\_stats), 8
- circular\_var(), 14, 25
- compact\_grid, 15
- compact\_grid(), 56, 83
- confidence, 15
- confidence\_angle (confidence), 15
- confidence\_angle(), 14
- confidence\_interval (confidence), 15
- confidence\_interval(), 77
- confidence\_interval\_fisher, 17
- confidence\_interval\_fisher(), 62, 77
- conjugate\_Q4, 18
- coordinate\_mod, 21
- coordinates, 19
- coordinates2, 20

- cpm\_models, 21
- deg2rad (angle-conversion), 4
- deviation\_norm, 22
- deviation\_shmax, 23
- deviation\_shmax(), 42, 55
- dispersion, 24
- dist\_greatcircle, 28
- dist\_greatcircle(), 38, 83
- distance\_from\_pb, 26
- distance\_from\_pb(), 33, 62, 84, 85
- distance\_mod, 27
- distroll\_circstats (rolling\_test\_dist), 69
- distroll\_confidence (rolling\_test\_dist), 69
- distroll\_dispersion (rolling\_test\_dist), 69
- distroll\_dispersion\_sde (rolling\_test\_dist), 69
- download\_WSM2016 (import\_WSM2016), 34
- download\_WSM2016(), 87
- dvm (vonmises), 93
- dvm(), 49
- earth\_radius, 29
- earth\_radius(), 4
- equivalent\_rotation, 29
- equivalent\_rotation(), 34, 90
- est.kappa, 30
- euler\_pole, 31
- euler\_pole(), 66
- euler\_to\_Q4, 31
- eulerpole\_greatcircles (stress\_paths), 87
- eulerpole\_loxodromes (stress\_paths), 87
- eulerpole\_paths (stress\_paths), 87
- eulerpole\_smallcircles (stress\_paths), 87
- geographical\_to\_cartesian (coordinates), 19
- geographical\_to\_cartesian(), 20
- geographical\_to\_PoR (por\_transformation\_df), 57
- geographical\_to\_PoR\_quat (por\_transformation\_quat), 57
- geographical\_to\_PoR\_raster (raster\_transformation), 63
- geographical\_to\_PoR\_sf (por\_transformation\_sf), 58
- geographical\_to\_spherical (coordinates), 19
- get\_azimuth, 32
- get\_azimuth(), 32
- get\_distance, 33
- get\_projected\_pb\_strike, 33
- get\_relrot, 34
- graphics::arrows(), 6
- graphics::polygon(), 75
- graphics::segments(), 75
- haversine (spherical\_angle), 80
- haversine(), 28
- iceland (stress\_data), 85
- import\_WSM2016, 34
- is.euler, 36
- is.Q4, 36
- kernel\_dispersion, 37
- kernel\_dispersion(), 15
- kuiper\_test, 38
- kuiper\_test(), 64
- latitude\_modulo (coordinate\_mod), 21
- line\_azimuth, 39
- lines\_azimuths (line\_azimuth), 39
- load\_WSM2016 (import\_WSM2016), 34
- load\_WSM2016(), 35
- longitude\_modulo (coordinate\_mod), 21
- mean\_resultant\_length, 40
- mean\_resultant\_length(), 13, 16, 64
- mean\_SC, 41
- model\_shmax, 42
- model\_shmax(), 24, 55, 59, 89
- norm\_chisq, 43
- norm\_chisq(), 38, 55, 62, 64, 85
- normalize\_Q4, 43
- nuvel1, 44
- nuvel1\_plates, 45
- orthodrome (spherical\_angle), 80
- orthodrome(), 28, 59
- parse\_wsm, 46
- parse\_wsm\_quality (parse\_wsm), 46

- pb2002, 46
- plates, 47
- plot\_density, 48
- plot\_points, 49
- PoR2Geo\_azimuth, 50
- PoR\_coordinates, 51
- PoR\_crs, 52
- PoR\_map, 53
- PoR\_shmax, 54
- PoR\_shmax(), 42, 51, 53, 59, 62, 85
- PoR\_stress2grid, 55
- PoR\_stress2grid(), 15, 83
- PoR\_to\_geographical
  - (por\_transformation\_df), 57
- PoR\_to\_geographical\_quat
  - (por\_transformation\_quat), 57
- PoR\_to\_geographical\_raster
  - (raster\_transformation), 63
- PoR\_to\_geographical\_sf
  - (por\_transformation\_sf), 58
- por\_transformation\_df, 57
- por\_transformation\_quat, 57
- por\_transformation\_sf, 58
- prd\_err, 59
- product\_Q4, 60
- projected\_pb\_strike, 60
- projected\_pb\_strike(), 33
- pvm (vonmises), 93
  
- Q4\_to\_euler, 61
- quantise\_wsm\_quality (parse\_wsm), 46
- quaternion (relative\_rotation), 65
- quick\_plot, 62
- quick\_plot(), 85
- qvm (vonmises), 93
  
- rad2deg (angle-conversion), 4
- raster\_transformation, 63
- rayleigh\_test, 63
- rayleigh\_test(), 38, 96
- relative\_rotation, 65
- relative\_rotation(), 30
- roll\_circstats, 71
- roll\_confidence (rolling\_test), 66
- roll\_dispersion (rolling\_test), 66
- roll\_dispersion\_CI (rolling\_test), 66
- roll\_dispersion\_sde (rolling\_test), 66
- roll\_normchisq (rolling\_test), 66
- roll\_rayleigh (rolling\_test), 66
- rolling\_test, 66
- rolling\_test\_dist, 69
- rose, 73
- rose(), 77
- rose\_fan (rose\_geom), 75
- rose\_geom, 75
- rose\_line (rose\_geom), 75
- rose\_stats, 76
- rotation (relative\_rotation), 65
- rotation\_Q4, 77
- rvm (vonmises), 93
  
- sample\_circular\_dispersion
  - (circle\_stats), 8
- sample\_circular\_dispersion(), 17
- sample\_median, 78
- san\_andreas (stress\_data), 85
- second\_central\_moment, 79
- second\_central\_moment(), 14
- sf::st\_crs(), 52
- smoothr::densify(), 26, 60
- spatstat.explore::rose(), 74
- spec\_atan, 80
- spherical\_angle, 80
- spherical\_to\_cartesian (coordinates2),
  - 20
- spherical\_to\_cartesian(), 19
- spherical\_to\_geographical
  - (coordinates2), 20
- stress2grid, 81
- stress2grid(), 15, 56
- stress\_analysis, 84
- stress\_colors, 85
- stress\_data, 85
- stress\_paths, 87
- superimposed\_shmax, 88
- superimposed\_shmax(), 90
- superimposed\_shmax\_PB, 89
- superimposed\_shmax\_PB(), 89
  
- tectonic.colors, 90
- tectonic.colors(), 53
- tibet (stress\_data), 85
  
- vcross, 91
- vincenty (spherical\_angle), 80
- vincenty(), 28
- vm\_qqplot, 92
- vonmises, 93

watson\_test, [94](#)  
watson\_test(), [64](#)  
weighted\_rayleigh, [95](#)  
weighted\_rayleigh(), [62](#), [64](#)  
  
zoo::na.fill(), [68](#), [72](#)  
zoo::rollapply(), [68](#), [72](#)