Package: rotations (via r-universe)

October 28, 2026

Type Package

Title Working with Rotation Data

Version 1.6.7

Description Tools for working with rotational data, including simulation from the most commonly used distributions on SO(3), methods for different Bayes, mean and median type estimators for the central orientation of a sample, confidence/credible regions for the central orientation based on those estimators and a novel visualization technique for rotation data. Most recently, functions to identify potentially discordant (outlying) values have been added. References: Bingham, Melissa A. and Nordman, Dan J. and Vardeman, Steve B. (2009), Bingham, Melissa A and Vardeman, Stephen B and Nordman, Daniel J (2009), Bingham, Melissa A and Nordman, Daniel J and Vardeman, Stephen B (2010), Leon, C.A. and Masse, J.C. and Rivest, L.P. (2006), Hartley, R and Aftab, K and Trumpf, J. (2011), Stanfill, Bryan and Genschel, Ulrike and Hofmann, Heike (2013), Maonton, Jonathan (2004), Mardia, KV and Jupp, PE (2000, ISBN:9780471953333), Rancourt, D. and Rivest, L.P. and Asselin, J. (2000), Chang, Ted and Rivest, Louis-Paul (2001), Fisher, Nicholas I. (1996, ISBN:0521568900).

License MIT + file LICENSE

Depends R (>= 4.1)

Imports ggplot2, gridExtra, Rcpp

Suggests knitr, onion, orientlib, testthat, rmarkdown

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/stanfill/rotationsC

BugReports https://github.com/stanfill/rotationsC/issues

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Collate 'Q4-class.R' 'RcppExports.R' 'preliminary.R' 'SO3-class.R' 'bayes.R' 'data.R' 'distributions.R' 'estimators.R' 'grid-search.R' 'help.R' 'kappa.R' 'plot.R' 'primitives.R' 'regions.R' 'robust.R' 'rotations-package.R' 'zzz.R'

NeedsCompilation yes

Author Bryan Stanfill [aut, cre], Heike Hofmann [aut], Ulrike Genschel [aut], Aymeric Stamm [ctb] (ORCID: <https://orcid.org/0000-0002-8725-3654>), Luciano Selzer [ctb]

Maintainer Bryan Stanfill <bstanfill2003@gmail.com>

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-07-10 16:20:03 UTC

RemoteUrl https://github.com/cran/rotations

RemoteRef HEAD

RemoteSha 318c5ee276eecc7d40879c423f22d8120e07ee3a

Contents

Angular-distributions . . . 3
Arithmetic . . . 4
bayes.mean . . . 5
bayesCR . . . 7
Cayley . . . 8
cayley.kappa . . . 9
center . . . 10
chang . . . 11
discord . . . 12
drill . . . 13
Fisher . . . 14
fisher.kappa . . . 16
fisheretal . . . 17
genR . . . 18
gradient.search . . . 19
Haar . . . 20
head . . . 21
log.SO3 . . . 24
Maxwell . . . 25
maxwell.kappa . . . 26
MCMCSO3 . . . 27
mean . . . 28
median . . . 30
mis.angle . . . 31
mis.axis . . . 33
Mises . . . 34
nickel . . . 35

plot . . . 37
pointsXYZ . . . 39
prentice . . . 40
project.SO3 . . . 41
Q4 . . . 42
region . . . 43
rot.dist . . . 45
rotations . . . 46
rotdist.sum . . . 47
skew.exp . . . 48
SO3 . . . 49
tail . . . 51
UARS . . . 53
vmises.kappa . . . 55
weighted.mean . . . 56
zhang . . . 57

Index **59**

`Angular-distributions` *Angular distributions*

Description

Density, distribution function and random variate generation for symmetric probability distributions in the rotations package.

Details

The functions for the density function and random variate generation are named in the usual form dxxxx, pxxxx and rxxxx, respectively.

- See `Cayley` for the Cayley distribution.
- See `Fisher` for the matrix Fisher distribution.
- See `Haar` for the uniform distribution on the circle.
- See `Maxwell` for the Maxwell-Boltzmann distribution on the circle.
- See `Mises` for the von Mises-Fisher distribution.

`Arithmetic` *Arithmetic operators on SO(3)*

Description

These binary operators perform arithmetic on rotations in quaternion or rotation matrix form (or objects which can be coerced into them).

Usage

```
## S3 method for class 'SO3'
x + y

## S3 method for class 'SO3'
x - y = NULL

## S3 method for class 'Q4'
x + y

## S3 method for class 'Q4'
x - y = NULL
```

Arguments

x	first argument
y	second argument (optional for subtraction)

Details

The rotation group SO(3) is a multiplicative group so "adding" rotations R_1 and R_2 results in $R_1 + R_2 = R_2 R_1$. Similarly, the difference between rotations R_1 and R_2 is $R_1 - R_2 = R_2^\top R_1$. With this definition it is clear that $R_1 + R_2 - R_2 = R_2^\top R_2 R_1 = R_1$. If only one rotation is provided to subtraction then the inverse (transpose) it returned, e.g. $-R_2 = R_2^\top$.

Value

+	the result of rotating the identity frame through x then y
-	the difference of the rotations, or the inverse rotation of only one argument is provided

Examples

```
U <- c(1, 0, 0)          #Rotate about the x-axis
R1 <- as.SO3(U, pi/8)    #Rotate pi/8 radians about the x-axis
R2 <- R1 + R1            #Rotate pi/8 radians about the x-axis twice
mis.axis(R2)             #x-axis: (1,0,0)
mis.angle(R2)            #pi/8 + pi/8 = pi/4
```

```
R3 <- R1 - R1            #Rotate pi/8 radians about x-axis then back again
R3                       #Identity matrix

R4 <- -R1                #Rotate in the opposite direction through pi/8
R5 <- as.SO3(U, -pi/8)   #Equivalent to R4

M1 <- matrix(R1, 3, 3)   #If element-wise addition is requred,
M2 <- matrix(R2, 3, 3)   #translate them to matrices then treat as usual
M3 <- M1 + M2

M1 %*% M1                #Equivalent to R2
t(M1) %*% M1             #Equivalent to R3
t(M1)                    #Equivalent to R4 and R5

#The same can be done with quaternions: the identity rotation is (1, 0, 0, 0)
#and the inverse rotation of Q=(a, b, c, d) is -Q=(a, -b, -c, -d)

Q1 <- as.Q4(R1)
Q2 <- Q1 + Q1
mis.axis(Q2)
mis.angle(Q2)

Q1 - Q1                  #id.Q4 = (1, 0, 0, 0)
```

`bayes.mean` *Parameter estimates based on non-informative Bayes*

Description

Use non-informative Bayes to estimate the central orientation and concentration parameter of a sample of rotations.

Usage

```
bayes.mean(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)

## S3 method for class 'SO3'
bayes.mean(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)

## S3 method for class 'Q4'
bayes.mean(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)
```

Arguments

`x`	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
`type`	Angular distribution assumed on R. Options are `Cayley`, `Fisher` or `Mises`
`S0`	initial estimate of central orientation

kappa0	initial estimate of concentration parameter
tuneS	central orientation tuning parameter, concentration of proposal distribution
tuneK	concentration tuning parameter, standard deviation of proposal distribution
burn_in	number of draws to use as burn-in
m	number of draws to keep from posterior distribution

Details

The procedures detailed in *bingham2009b* and *bingham2010* are implemented to obtain draws from the posterior distribution for the central orientation and concentration parameters for a sample of 3D rotations. A uniform prior on SO(3) is used for the central orientation and the Jeffreys prior determined by type is used for the concentration parameter.

bingham2009b bingham2010

Value

list of

- Shat Mode of the posterior distribution for the central orientation S
- kappa Mean of the posterior distribution for the concentration kappa

See Also

mean.SO3, median.SO3

Examples

```
Rs <- ruars(20, rvmises, kappa = 10)

Shat <- mean(Rs)              #Estimate the central orientation using the projected mean
rotdist.sum(Rs, Shat, p = 2)   #The projected mean minimizes the sum of squared Euclidean
rot.dist(Shat)                 #distances, compute the minimized sum and estimator bias

#Estimate the central orientation using the posterior mode (not run due to time constraints)
#Compare it to the projected mean in terms of the squared Euclidean distance and bias

ests <- bayes.mean(Rs, type = "Mises", S0 = mean(Rs), kappa0 = 10, tuneS = 5000,
                   tuneK = 1, burn_in = 1000, m = 5000)

Shat2 <- ests$Shat             #The posterior mode is the 'Shat' object
rotdist.sum(Rs, Shat2, p = 2)  #Compute sum of squared Euclidean distances
rot.dist(Shat2)                #Bayes estimator bias
```

`bayesCR` *Bayes credible regions*

Description

Find the radius of a $100(1-\alpha)\%$ credible region for the central orientation and concentration parameter using non-informative Bayesian methods.

Usage

```
bayesCR(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000, alp = 0.1)

## S3 method for class 'SO3'
bayesCR(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000, alp = 0.1)

## S3 method for class 'Q4'
bayesCR(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000, alp = 0.1)
```

Arguments

`x`	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
`type`	Angular distribution assumed on R. Options are `Cayley`, `Fisher` or `Mises`
`S0`	initial estimate of central orientation
`kappa0`	initial estimate of concentration parameter
`tuneS`	central orientation tuning parameter, concentration of proposal distribution
`tuneK`	concentration tuning parameter, standard deviation of proposal distribution
`burn_in`	number of draws to use as burn-in
`m`	number of draws to keep from posterior distribution
`alp`	alpha level desired, e.g. 0.05 or 0.10.

Details

Compute the radius of a $100(1-\alpha)\%$ credible region for the central orientation and concentration parameter as described in *bingham2009b* and *bingham2010*. The posterior mode is returned along with the radius of the credible region centered at the posterior mode.

bingham2009b bingham2010

Value

list of

- `Shat,Qhat` Mode of the posterior distribution for the central orientation S
- `Radius` Radius of the credible region centered at the posterior mode

See Also

fisheretal, prentice, chang, zhang

Examples

```
#Not run due to time constraints

Rs <- ruars(20, rvmises, kappa = 10)

#Compare the region size of the moment based theory mean estimator to the
#Bayes region.

region(Rs, method = "direct", type = "theory", estimator = "mean", alp=0.1, m = 100)
bayesCR <- region(Rs, type = "Mises", method = "Bayes", estimator = "mean", S0 = mean(Rs),
                kappa0 = 10, tuneS = 5000, tuneK = 1, burn_in = 1000, alp = .01, m = 5000)

bayesCR$Radius       #Region size is give by "Radius"
bayesCR$Shat         #The Bayes region is centered around the posterior mode: "Shat"
```

Cayley *The symmetric Cayley distribution*

Description

Density, distribution function and random generation for the Cayley distribution with concentration kappa κ.

Usage

```
dcayley(r, kappa = 1, nu = NULL, Haar = TRUE)

pcayley(q, kappa = 1, nu = NULL, lower.tail = TRUE)

rcayley(n, kappa = 1, nu = NULL)
```

Arguments

r, q	vector of quantiles.
kappa	concentration parameter.
nu	circular variance, can be used in place of kappa.
Haar	logical; if TRUE density is evaluated with respect to the Haar measure.
lower.tail	logical; if TRUE (default) probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
n	number of observations. If length(n)>1, the length is taken to be the number required.

Details

The symmetric Cayley distribution with concentration κ has density

$$C_C(r|\kappa) = \frac{1}{\sqrt{\pi}} \frac{\Gamma(\kappa+2)}{\Gamma(\kappa+1/2)} 2^{-(\kappa+1)} (1+\cos r)^{\kappa} (1-\cos r).$$

The Cayley distribution is equivalent to the de la Vallee Poussin distribution of *Schaeben1997*.

Schaeben1997 leon2006

Value

`dcayley`	gives the density
`pcayley`	gives the distribution function
`rcayley`	generates a vector of random deviates

See Also

Angular-distributions for other distributions in the rotations package.

Examples

```
r <- seq(-pi, pi, length = 500)

#Visualize the Cayley density fucntion with respect to the Haar measure
plot(r, dcayley(r, kappa = 10), type = "l", ylab = "f(r)")

#Visualize the Cayley density fucntion with respect to the Lebesgue measure
plot(r, dcayley(r, kappa = 10, Haar = FALSE), type = "l", ylab = "f(r)")

#Plot the Cayley CDF
plot(r,pcayley(r,kappa = 10), type = "l", ylab = "F(r)")

#Generate random observations from Cayley distribution
rs <- rcayley(20, kappa = 1)
hist(rs, breaks = 10)
```

`cayley.kappa` *Circular variance and concentration parameter*

Description

Return the concentration parameter that corresponds to a given circular variance.

Usage

```
cayley.kappa(nu)
```

Arguments

nu	circular variance

Details

The concentration parameter κ does not translate across circular distributions. A commonly used measure of spread in circular distributions that does translate is the circular variance defined as $\nu = 1 - E[\cos(r)]$ where $E[\cos(r)]$ is the mean resultant length. See *mardia2000* for more details. This function translates the circular variance ν into the corresponding concentration parameter κ for the Cayley distribution.

mardia2000

Value

Concentration parameter corresponding to nu.

See Also

Cayley

Examples

```
# Find the concentration parameter for circular variances 0.25, 0.5, 0.75
cayley.kappa(0.25)
cayley.kappa(0.5)
cayley.kappa(0.75)
```

---

center *Center rotation data*

Description

This function will take the sample Rs and return the sample Rs centered at S. That is, the ith observation of Rs denoted R_i is returned as $S^\top R_i$. If S is the true center then the projected mean should be close to the 3-by-3 identity matrix.

Usage

```
center(x, S)

## S3 method for class 'SO3'
center(x, S)

## S3 method for class 'Q4'
center(x, S)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
S	the rotation or a matrix of $n \times p$ rotations about which to center each row of x.

Value

The sample centered about S

Examples

```
Rs <- ruars(5, rcayley)
cRs <- center(Rs, mean(Rs))
mean(cRs)                     #Close to identity matrix

all.equal(cRs, Rs - mean(Rs))  #TRUE, center and '-' have the same effect
                               #See ?"-.SO3" for more details

center(Rs,Rs)            #n-Identity matrices: If the second argument is of the same dimension
                               #as Rs then each row is centered around the corresponding
                               #row in the first argument
```

---

chang *M-estimator asymptotic confidence region*

Description

Compute the radius of a $100(1 - \alpha)\%$ confidence region for the central orientation based on M-estimation theory.

Usage

```
chang(x, estimator, alp = NULL)

## S3 method for class 'SO3'
chang(x, estimator, alp = NULL)

## S3 method for class 'Q4'
chang(x, estimator, alp = NULL)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
estimator	character string either "mean" or "median."
alp	alpha level desired, e.g. 0.05 or 0.10.

Details

Compute the radius of a $100(1-\alpha)\%$ confidence region for the central orientation centered at the projected mean or median based on a result due to *chang2001* among others. By construction each axis will have the same radius so the radius reported is for all three axes. This method is called "direct" because it uses M-estimation theory for SO(3) directly instead of relying on the transformation of a result from directional statistics like `prentice` and `fisheretal` do.

chang2001

Value

Radius of the confidence region centered at the specified estimator.

See Also

`bayesCR`, `prentice`, `fisheretal`, `zhang`

Examples

```
Rs <- ruars(20, rcayley, kappa = 100)

# The chang method can be accesed from the "region" function or the "chang" function
region(Rs, method = "direct", type = "asymptotic", alp = 0.1, estimator = "mean")
chang(Rs, estimator = "mean", alp = 0.1)
```

---

discord *Measure of Discord*

Description

This function computes a measure of discord for a sample of random rotations. The larger the statistic value the less likely it is the corresponding observation was generated by the same mechanism the rest of the data as generated by. It can be used to test for outliers in SO(3) by comparing it to an F distribution with 3,3(n-2) df for the Cayley or matrix Fisher distributions or to an F distribution with 1,n-2 df for the von Mises Fisher distribution.

Usage

```
discord(x, type, t = 1L, obs = 1:nrow(x))
```

Arguments

`x`	The sample of random rotations
`type`	To specify if "intrinsic" or "extrinsic" approach should be used to compute the statistic
`t`	If test blocs then the bloc size, set to 1 by default
`obs`	integer vector specifying which observation(s) to compute the measure of discord for

Value

The Hi statistic for each group of size t is returned. If t>1 then which observations that define each group of size t is returned as well.

Examples

```
#Compute the measures of discord for a sample from the Cayley distribution
# Intrinsic examples are commented out but are below if you're interested

Rss <- ruars(20,rcayley,kappa=1)
Hi <- discord(Rss, type='intrinsic')
He <- discord(Rss, type='extrinsic')

#Compare to the theoretical F distribution
OrdHi <- sort(Hi)
OrdHe <- sort(He)

par(mfrow=c(1,2))
plot(ecdf(OrdHi),main='Intrinsic',xlim=range(c(OrdHi,OrdHe)))
lines(OrdHi,pf(OrdHi,3,3*(length(OrdHi)-2)))

plot(ecdf(OrdHe),main='Extrinsic',xlim=range(c(OrdHi,OrdHe)))
lines(OrdHi,pf(OrdHi,3,3*(length(OrdHe)-2)))
layout(1)
```

---

drill *Drill data set*

Description

The drill data set was collected to assess variation in human movement while performing a task (Rancourt, 1995). Eight subjects drilled into a metal plate while being monitored by infared cameras. Quaternions are used to represent the orientation of each subjects' wrist, elbow and shoulder in one of six positions. For some subjects several replicates are available. See Rancourt et al. (2000) for one approach to analyzing these data.

Usage

```
drill
```

Format

A data frame with 720 observations on the following 8 variables:

`Subject` Subject number (1-8)

`Joint` Joint name (Wrist, elbow, shoulder)

`Position` Drilling position (1-6)

`Replicate` Replicate number (1-5)

- `Q1` First element of orientation (quaternion)
- `Q2` Second element of orientation (quaternion)
- `Q3` Third element of orientation (quaternion)
- `Q4` Fourth element of orientation (quaternion)

Source

https://louis-paul-rivest.fsg.ulaval.ca/en

References

1. Rancourt, D. (1995). "Arm posture and hand mechanical impedance in the control of a hand-held power drill." Ph.D. Thesis, MIT.
2. Rancourt, D., Rivest, L. & Asselin, J. (2000). "Using orientation statistics to investigate variations in human kinematics." Journal of the Royal Statistical Society: Series C (Applied Statistics), 49(1), pp. 81-94.

Examples

```
# Estimate central orientation of the first subject's wrist
Subject1Wrist <- subset(drill, Subject == 1 & Joint == "Wrist")
Qs <- as.Q4(Subject1Wrist[, 5:8])
mean(Qs)


  # Plot Subject 1's wrist measurements using the connection to rotation matrices
  plot(Qs, col = c(1, 2, 3))


# Translate the quaternion measurements into rotations and
# estimate the central orientation in terms of rotations
Rs <- as.SO3(Qs)
mean(Rs)
```

`Fisher` *The matrix-Fisher distribution*

Description

Density, distribution function and random generation for the matrix-Fisher distribution with concentration `kappa` κ.

Usage

```
dfisher(r, kappa = 1, nu = NULL, Haar = TRUE)

pfisher(q, kappa = 1, nu = NULL, lower.tail = TRUE)

rfisher(n, kappa = 1, nu = NULL)
```

Arguments

r, q	vector of quantiles.
kappa	concentration parameter.
nu	circular variance, can be used in place of kappa.
Haar	logical; if TRUE density is evaluated with respect to the Haar measure.
lower.tail	logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
n	number of observations. If length(n)>1, the length is taken to be the number required.

Details

The matrix-Fisher distribution with concentration κ has density

$$C_{\mathrm{F}}(r|\kappa) = \frac{1}{2\pi[\mathrm{I}_0(2\kappa) - \mathrm{I}_1(2\kappa)]} e^{2\kappa\cos(r)}[1 - \cos(r)]$$

with respect to Lebesgue measure where $\mathrm{I}_p(\cdot)$ denotes the Bessel function of order p defined as $\mathrm{I}_p(\kappa) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(pr)e^{\kappa\cos r}dr$. If kappa>354 then random deviates are generated from the Cayley distribution because they agree closely for large kappa and generation is more stable from the Cayley distribution.

For large κ, the Bessel functon gives errors so a large κ approximation to the matrix-Fisher distribution is used instead, which is the Maxwell-Boltzmann density.

Value

dfisher	gives the density
pfisher	gives the distribution function
rfisher	generates random deviates

See Also

Angular-distributions for other distributions in the rotations package.

Examples

```
r <- seq(-pi, pi, length = 500)

#Visualize the matrix Fisher density fucntion with respect to the Haar measure
plot(r, dfisher(r, kappa = 10), type = "l", ylab = "f(r)")

#Visualize the matrix Fisher density fucntion with respect to the Lebesgue measure
plot(r, dfisher(r, kappa = 10, Haar = FALSE), type = "l", ylab = "f(r)")

#Plot the matrix Fisher CDF
plot(r,pfisher(r,kappa = 10), type = "l", ylab = "F(r)")

#Generate random observations from matrix Fisher distribution
rs <- rfisher(20, kappa = 1)
hist(rs, breaks = 10)
```

`fisher.kappa` *Circular variance and concentration parameter*

Description

Return the concentration parameter that corresponds to a given circular variance.

Usage

```
fisher.kappa(nu)
```

Arguments

nu	circular variance

Details

The concentration parameter κ does not translate across circular distributions. A commonly used measure of spread in circular distributions that does translate is the circular variance defined as $\nu = 1 - E[\cos(r)]$ where $E[\cos(r)]$ is the mean resultant length. See *mardia2000* for more details. This function translates the circular variance ν into the corresponding concentration parameter κ for the matrix-Fisher distribution. For numerical stability, a maximum κ of 350 is returned.

mardia2000

Value

Concentration parameter corresponding to nu.

See Also

`Fisher`

Examples

```
# Find the concentration parameter for circular variances 0.25, 0.5, 0.75
fisher.kappa(0.25)
fisher.kappa(0.5)
fisher.kappa(0.75)
```

`fisheretal` *Transformation based pivotal bootstrap confidence region*

Description

Find the radius of a $100(1-\alpha)\%$ confidence region for the central orientation based on transforming a result from directional statistics.

Usage

```
fisheretal(x, alp = NULL, boot = TRUE, m = 300, symm = TRUE)

## S3 method for class 'Q4'
fisheretal(x, alp = NULL, boot = TRUE, m = 300, symm = TRUE)

## S3 method for class 'SO3'
fisheretal(x, alp = NULL, boot = TRUE, m = 300, symm = TRUE)
```

Arguments

`x`	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
`alp`	alpha level desired, e.g. 0.05 or 0.10.
`boot`	should the bootstrap or normal theory critical value be used.
`m`	number of bootstrap replicates to use to estimate critical value.
`symm`	logical; if TRUE (default), a symmetric region is constructed.

Details

Compute the radius of a $100(1-\alpha)\%$ confidence region for the central orientation based on the projected mean estimator using the method for the mean polar axis as proposed in *fisher1996*. To be able to reduce their method to a radius requires the additional assumption of rotational symmetry, equation (10) in *fisher1996*.

fisher1996

Value

Radius of the confidence region centered at the projected mean.

See Also

`bayesCR`, `prentice`, `chang`, `zhang`

Examples

```
Qs<-ruars(20, rcayley, kappa = 100, space = 'Q4')

# The Fisher et al. method can be accesed from the "region" function or the "fisheretal" function
region(Qs, method = "transformation", type = "bootstrap", alp = 0.1,
symm = TRUE, estimator = "mean")
fisheretal(Qs, alp = 0.1, boot = TRUE, symm = TRUE)
```

---

genR *Generate rotations*

Description

Generate rotations in matrix format using Rodrigues' formula or quaternions.

Usage

```
genR(r, S = NULL, space = "SO3")
```

Arguments

r	vector of angles.
S	central orientation.
space	indicates the desired representation: rotation matrix "SO3" or quaternions "Q4."

Details

Given a vector $U = (u_1, u_2, u_3)^\top \in R^3$ of length one and angle of rotation r, a 3×3 rotation matrix is formed using Rodrigues' formula

$$\cos(r) I_{3\times 3} + \sin(r)\Phi(U) + (1 - \cos(r))UU^\top$$

where $I_{3\times 3}$ is the 3×3 identity matrix, $\Phi(U)$ is a 3×3 skew-symmetric matrix with upper triangular elements $-u_3$, u_2 and $-u_1$ in that order.

For the same vector and angle a quaternion is formed according to

$$q = [cos(\theta/2), sin(\theta/2)U]^\top.$$

Value

A $n \times p$ matrix where each row is a random rotation matrix ($p = 9$) or quaternion ($p = 4$).

Examples

```
r <- rvmises(20, kappa = 0.01)
Rs <- genR(r, space = "SO3")
Qs <- genR(r, space = "Q4")
```

`gradient.search` *Gradient optimization for rotation data*

Description

Gradient based optimization for user defined central orientation of a rotation sample.

Usage

```
gradient.search(
  sample,
  error,
  minerr = 1e-05,
  start = mean(sample),
  theta = NULL
)
```

Arguments

`sample`	sample of rotations.
`error`	user defined function to observed distance between sample and estimate, has to have parameters for the sample and the estimate.
`minerr`	minimal distance to consider for convergence.
`start`	starting value for the estimation.
`theta`	size of the grid considered.

Value

list of

- `Shat` estimate of the main direction
- `iter` number of iterations necessary for convergence
- `theta` final size of the grid
- `minerr` error used for convergence
- `error` numeric value of total sample's distance from main direction

Examples

```
# minimize L1 norm:
L1.error <- function(sample, Shat) {
  sum(rot.dist(sample, Shat, method = "intrinsic", p = 1))
}

cayley.sample <- ruars(n = 10, rangle = rcayley, nu = 1, space = 'SO3')
SL1 <- gradient.search(cayley.sample, L1.error, start = id.SO3)
```

```
# visually no perceptible difference between median estimates from in-built function and
# gradient based search (for almost all starting values)


plot(cayley.sample, center=SL1$Shat, show_estimates="all")
```

Haar *Uniform distribution*

Description

Density, distribution function and random generation for the uniform distribution.

Usage

```
dhaar(r)

phaar(q, lower.tail = TRUE)

rhaar(n)
```

Arguments

r, q	vector of quantiles.
lower.tail	logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
n	number of observations. If length(n)>1, the length is taken to be the number required.

Details

The uniform distribution has density

$$C_U(r) = \frac{[1 - cos(r)]}{2\pi}$$

with respect to the Lebesgue measure. The Haar measure is the volume invariant measure for SO(3) that plays the role of the uniform measure on SO(3) and $C_U(r)$ is the angular distribution that corresponds to the uniform distribution on SO(3), see UARS. The uniform distribution with respect to the Haar measure is given by

$$C_U(r) = \frac{1}{2\pi}.$$

Because the uniform distribution with respect to the Haar measure gives a horizontal line at 1 with respect to the Lebesgue measure, we called this distribution 'Haar.'

Value

dhaar	gives the density
phaar	gives the distribution function
rhaar	generates random deviates

See Also

Angular-distributions for other distributions in the rotations package.

Examples

```
r <- seq(-pi, pi, length = 1000)

#Visualize the uniform distribution with respect to Lebesgue measure
plot(r, dhaar(r), type = "l", ylab = "f(r)")

#Visualize the uniform distribution with respect to Haar measure, which is
#a horizontal line at 1
plot(r, 2*pi*dhaar(r)/(1-cos(r)), type = "l", ylab = "f(r)")

#Plot the uniform CDF
plot(r,phaar(r), type = "l", ylab = "F(r)")

#Generate random observations from uniform distribution
rs <- rhaar(50)

#Visualize on the real line
hist(rs, breaks = 10)
```

---

head *Return the First or Last Parts of an Object*

Description

Returns the first or last parts of a vector, matrix, array, table, data frame or function. Since `head()` and `tail()` are generic functions, they have been extended to other classes, including `"ts"` from **stats**.

Usage

```
## S3 method for class 'SO3'
head(x, n = 6L, ...)

## S3 method for class 'Q4'
head(x, n = 6L, ...)
```

Arguments

x	an object
n	an integer vector of length up to `dim(x)` (or 1, for non-dimensioned objects). A `logical` is silently coerced to integer. Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive

value of `n[i]` includes the first/last `n[i]` indices in that dimension, while a negative value excludes the last/first `abs(n[i])`, including all remaining indices. `NA` or non-specified values (when `length(n) < length(dim(x))`) select all indices in that dimension. Must contain at least one non-missing value.

`...` arguments to be passed to or from other methods.

Details

For vector/array based objects, `head()` (`tail()`) returns a subset of the same dimensionality as `x`, usually of the same class. For historical reasons, by default they select the first (last) 6 indices in the first dimension ("rows") or along the length of a non-dimensioned vector, and the full extent (all indices) in any remaining dimensions. `head.matrix()` and `tail.matrix()` are exported.

The default and array(/matrix) methods for `head()` and `tail()` are quite general. They will work as is for any class which has a `dim()` method, a `length()` method (only required if `dim()` returns `NULL`), and a `[` method (that accepts the `drop` argument and can subset in all dimensions in the dimensioned case).

For functions, the lines of the deparsed function are returned as character strings.

When `x` is an array(/matrix) of dimensionality two and more, `tail()` will add dimnames similar to how they would appear in a full printing of `x` for all dimensions `k` where `n[k]` is specified and non-missing and `dimnames(x)[[k]]` (or `dimnames(x)` itself) is `NULL`. Specifically, the form of the added dimnames will vary for different dimensions as follows:

`k=1` **(rows):** `"[n,]"` (right justified with whitespace padding)

`k=2` **(columns):** `"[,n]"` (with *no* whitespace padding)

`k>2` **(higher dims):** `"n"`, i.e., the indices as *character* values

Setting `keepnums = FALSE` suppresses this behaviour.

As `data.frame` subsetting ('indexing') keeps `attributes`, so do the `head()` and `tail()` methods for data frames.

The auxiliary function `.checkHT(d, n)` is useful in `head(x, n)` or `tail(x, n)` methods, checking validity of `d <- dim(x)` and `n`.

Value

An object (usually) like `x` but generally smaller. Hence, for `array`s, the result corresponds to `x[..., drop=FALSE]`. For `ftable` objects `x`, a transformed `format(x)`.

Note

For array inputs the output of `tail` when `keepnums` is `TRUE`, any dimnames vectors added for dimensions `>2` are the original numeric indices in that dimension *as character vectors*. This means that, e.g., for 3-dimensional array `arr`, `tail(arr, c(2,2,-1))[ , , 2]` and `tail(arr, c(2,2,-1))[ , , "2"]` may both be valid but have completely different meanings.

Author(s)

Patrick Burns, improved and corrected by R-Core. Negative argument added by Vincent Goulet. Multi-dimension support added by Gabriel Becker.

Examples

```
head(letters)
head(letters, n = -6L)

head(freeny.x, n = 10L)
head(freeny.y)

head(gait) # 3d array
head(gait, c(6L, 2L))
head(gait, c(6L, 2L, -1L))

tail(letters)
tail(letters, n = -6L)

tail(freeny.x)
## the bottom-right "corner" :
tail(freeny.x, n = c(4, 2))
tail(freeny.y)

tail(gait)
tail(gait, c(6L, 2L))
tail(gait, c(6L, 2L, -1L))

## gait without dimnames --> keepnums showing original row/col numbers
a3 <- gait ; dimnames(a3) <- NULL
tail(a3, c(6, 2, -1))# keepnums = TRUE is default here!
tail(a3, c(6, 2, -1),  keepnums = FALSE)

## data frame w/ a (non-standard) attribute:
treeS <- structure(trees, foo = "bar")
(n <- nrow(treeS))
stopifnot(exprs = { # attribute is kept
    identical(htS <- head(treeS), treeS[1:6, ])
    identical(attr(htS, "foo") , "bar")
    identical(tlS <- tail(treeS), treeS[(n-5):n, ])
    ## BUT if I use "useAttrib(.)", this is *not* ok, when n is of length 2:
    ## --- because [i,j]-indexing of data frames *also* drops "other" attributes ..
    identical(tail(treeS, 3:2), treeS[(n-2):n, 2:3] )
})

tail(library) # last lines of function

head(stats::ftable(Titanic))

## 1d-array (with named dim) :
a1 <- array(1:7, 7); names(dim(a1)) <- "O2"
stopifnot(exprs = {
  identical( tail(a1, 10), a1)
  identical( head(a1, 10), a1)
  identical( head(a1, 1), a1 [1 , drop=FALSE] ) # was a1[1] in R <= 3.6.x
  identical( tail(a1, 2), a1[6:7])
  identical( tail(a1, 1), a1 [7 , drop=FALSE] ) # was a1[7] in R <= 3.6.x
```

```
})
```

`log.SO3` *Rotation logarithm*

Description

Compute the logarithm of a rotation matrix, which results in a 3×3 skew-symmetric matrix. This function maps the lie group $SO(3)$ into its tangent space, which is the space of all 3×3 skew symmetric matrices, the lie algebra $so(3)$. For details see e.g. *moakher02*.

Usage

```
## S3 method for class 'SO3'
log(x, ...)
```

Arguments

`x`	$n \times 9$ matrix where each row corresponds to a random rotation matrix.
`...`	additional arguments.

Details

moakher02

Value

Skew symmetric matrix $\log(R)$.

Examples

```
Rs <- ruars(20, rcayley)

#Here we demonstrate how the logarithm can be used to determine the angle and
#axis corresponding to the provided sample

lRs <- log(Rs)              #Take the logarithm of the sample
Ws <- lRs[,c(6, 7, 2)]      #The appropriate diagonal entries are the axis*angle
lens <- sqrt(rowSums(Ws^2))
axes <- mis.axis(Rs)
angs <- mis.angle(Rs)
all.equal(axes, Ws/lens)
all.equal(angs, lens)
```

`Maxwell` *The modified Maxwell-Boltzmann distribution*

Description

Density, distribution function and random generation for the Maxwell-Boltzmann distribution with concentration `kappa` κ restricted to the range $[-\pi, \pi)$.

Usage

```
dmaxwell(r, kappa = 1, nu = NULL, Haar = TRUE)

pmaxwell(q, kappa = 1, nu = NULL, lower.tail = TRUE)

rmaxwell(n, kappa = 1, nu = NULL)
```

Arguments

`r`, `q`	vector of quantiles.
`kappa`	concentration parameter.
`nu`	circular variance, can be used in place of `kappa`.
`Haar`	logical; if TRUE density is evaluated with respect to the Haar measure.
`lower.tail`	logical; if TRUE (default) probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
`n`	number of observations. If `length(n)>1`, the length is taken to be the number required.

Details

The Maxwell-Boltzmann distribution with concentration κ has density

$$C_{\mathrm{M}}(r|\kappa) = 2\kappa\sqrt{\frac{\kappa}{\pi}}r^2 e^{-\kappa r^2}$$

with respect to Lebesgue measure. The usual expression for the Maxwell-Boltzmann distribution can be recovered by setting $a = (2\kappa)^0.5$.

bingham2010

Value

`dmaxwell`	gives the density
`pmaxwell`	gives the distribution function
`rmaxwell`	generates a vector of random deviates

See Also

Angular-distributions for other distributions in the rotations package.

Examples

```
r <- seq(-pi, pi, length = 500)

#Visualize the Maxwell-Boltzmann density fucntion with respect to the Haar measure
plot(r, dmaxwell(r, kappa = 10), type = "l", ylab = "f(r)")

#Visualize the Maxwell-Boltzmann density fucntion with respect to the Lebesgue measure
plot(r, dmaxwell(r, kappa = 10, Haar = FALSE), type = "l", ylab = "f(r)")

#Plot the Maxwell-Boltzmann CDF
plot(r,pmaxwell(r,kappa = 10), type = "l", ylab = "F(r)")

#Generate random observations from Maxwell-Boltzmann distribution
rs <- rmaxwell(20, kappa = 1)
hist(rs, breaks = 10)
```

---

maxwell.kappa *Circular variance and concentration parameter*

Description

Return the concentration parameter that corresponds to a given circular variance.

Usage

```
maxwell.kappa(nu)
```

Arguments

nu	circular variance

Details

The concentration parameter κ does not translate across circular distributions. A commonly used measure of spread in circular distributions that does translate is the circular variance defined as $\nu = 1 - E[\cos(r)]$ where $E[\cos(r)]$ is the mean resultant length. See *mardia2000* for more details. This function translates the circular variance ν into the corresponding concentration parameter κ for the modified Maxwell-Boltzmann distribution. For numerical stability, a maximum κ of 1000 is returned.

Value

Concentration parameter corresponding to nu.

See Also

Maxwell

Examples

```
# Find the concentration parameter for circular variances 0.25, 0.5, 0.75
maxwell.kappa(0.25)
maxwell.kappa(0.5)
maxwell.kappa(0.75)
```

MCMCSO3 *MCMC for rotation data*

Description

Use non-informative Bayesian methods to infer about the central orientation and concentration parameter for a sample of rotations.

Usage

```
MCMCSO3(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)

## S3 method for class 'SO3'
MCMCSO3(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)

## S3 method for class 'Q4'
MCMCSO3(x, type, S0, kappa0, tuneS, tuneK, burn_in, m = 5000)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
type	Angular distribution assumed on R. Options are Cayley, Fisher or Mises
S0	initial estimate of central orientation
kappa0	initial estimate of concentration parameter
tuneS	central orientation tuning parameter, concentration of proposal distribution
tuneK	concentration tuning parameter, standard deviation of proposal distribution
burn_in	number of draws to use as burn-in
m	number of draws to keep from posterior distribution

Details

The procedures detailed in *bingham2009b* and *bingham2010* are implemented to obtain draws from the posterior distribution for the central orientation and concentration parameters for a sample of 3D rotations. A uniform prior on SO(3) is used for the central orientation and the Jeffreys prior determined by type is used for the concentration parameter.

bingham2009b bingham2010

Value

list of

- S Draws from the posterior distribution for central orientation S
- kappa Draws from the posterior distribution for concentration parameter kappa
- Saccept Acceptance rate for central orientation draws
- Kaccept Acceptance rate for concentration draws

Examples

```
#Not run due to time constraints

Rs <- ruars(20, rfisher, kappa = 10)
draws <- MCMCSO3(Rs, type = "Fisher", S0 = mean(Rs), kappa0 = 10, tuneS = 5000,
                 tuneK = 1,burn_in = 1000, m = 5000)
```

---

mean *Mean rotation*

Description

Compute the sample geometric or projected mean.

Usage

```
## S3 method for class 'SO3'
mean(x, type = "projected", epsilon = 1e-05, maxIter = 2000, ...)

## S3 method for class 'Q4'
mean(x, type = "projected", epsilon = 1e-05, maxIter = 2000, ...)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix form ($p = 9$) or quaternion ($p = 4$) form.
type	string indicating "projected" or "geometric" type mean estimator.
epsilon	stopping rule for the geometric-mean.
maxIter	maximum number of iterations allowed for geometric-mean.
...	additional arguments.

Details

This function takes a sample of 3D rotations (in matrix or quaternion form) and returns the projected arithmetic mean denoted $\widehat{\boldsymbol{S}}_P$ or geometric mean $\widehat{\boldsymbol{S}}_G$ according to the `type` option. For a sample of n rotations in matrix form $\boldsymbol{R}_i \in SO(3), i = 1, 2, \ldots, n$, the mean-type estimator is defined as

$$\widehat{\boldsymbol{S}} = argmin_{\boldsymbol{S} \in SO(3)} \sum_{i=1}^{n} d^2(\boldsymbol{R}_i, \boldsymbol{S})$$

where d is the Riemannian or Euclidean distance. For more on the projected mean see *moakher02* and for the geometric mean see *manton04*. For the projected mean from a quaternion point of view see *tyler1981*.

tyler1981, moakher02, manton04

Value

Estimate of the projected or geometric mean of the sample in the same parametrization.

See Also

`median.SO3`, `bayes.mean`, `weighted.mean.SO3`

Examples

```
Rs <- ruars(20, rvmises, kappa = 0.01)

# Projected mean
mean(Rs)

# Same as mean(Rs)
project.SO3(colMeans(Rs))

# Geometric mean
mean(Rs, type = "geometric")

# Bias of the projected mean
rot.dist(mean(Rs))

# Bias of the geometric mean
rot.dist(mean(Rs, type = "geometric"))

# Same thing with quaternion form
Qs <- as.Q4(Rs)
mean(Qs)
mean(Qs, type = "geometric")
rot.dist(mean(Qs))
rot.dist(mean(Qs, type = "geometric"))
```

median *Median rotation*

Description

Compute the sample projected or geometric median.

Usage

```
## S3 method for class 'SO3'
median(
  x,
  na.rm = FALSE,
  type = "projected",
  epsilon = 1e-05,
  maxIter = 2000,
  ...
)

## S3 method for class 'Q4'
median(
  x,
  na.rm = FALSE,
  type = "projected",
  epsilon = 1e-05,
  maxIter = 2000,
  ...
)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix form ($p = 9$) or quaternion ($p = 4$) form.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
type	string indicating "projected" or "geometric" type mean estimator.
epsilon	stopping rule.
maxIter	maximum number of iterations allowed before returning most recent estimate.
...	additional arguments.

Details

The median-type estimators are defined as

$$\widetilde{\boldsymbol{S}} = argmin_{\boldsymbol{S} \in SO(3)} \sum_{i=1}^{n} d(\boldsymbol{R}_i, \boldsymbol{S}).$$

If the choice of distance metric d is Riemannian then the estimator is called the geometric median, and if the distance metric in Euclidean then it is called the projected median. The algorithm used in the geometric case is discussed in *hartley11* and the projected case is in *stanfill2013*.

hartley11 stanfill2013

Value

Estimate of the projected or geometric median in the same parametrization.

See Also

`mean.SO3`, `bayes.mean`, `weighted.mean.SO3`

Examples

```
Rs <- ruars(20, rvmises, kappa = 0.01)

# Projected median
median(Rs)

# Geometric median
median(Rs, type = "geometric")

# Bias of the projected median
rot.dist(median(Rs))

# Bias of the geometric median
rot.dist(median(Rs, type = "geometric"))

Qs <- as.Q4(Rs)

# Projected median
median(Qs)

# Geometric median
median(Qs, type = "geometric")

# Bias of the projected median
rot.dist(median(Qs))

# Bias of the geometric median
rot.dist(median(Qs, type = "geometric"))
```

`mis.angle` *Misorientation angle*

Description

Compute the misorientation angle of a rotation.

Usage

```
mis.angle(x)

## S3 method for class 'SO3'
mis.angle(x)

## S3 method for class 'Q4'
mis.angle(x)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.

Details

Every rotation can be thought of as some reference coordinate system rotated about an axis through an angle. These quantities are referred to as the misorientation axis and misorientation angle, respectively, in the material sciences literature. This function returns the misorentation angle associated with a rotation assuming the reference coordinate system is the identity.

Value

Angle of rotation.

See Also

mis.axis

Examples

```
rs <- rcayley(20, kappa = 20)
Rs <- genR(rs, S = id.SO3)
mis.angle(Rs)

#If the central orientation is id.SO3 then mis.angle(Rs) and abs(rs) are equal
all.equal(mis.angle(Rs), abs(rs))  #TRUE

#For other reference frames, the data must be centered first
S <- genR(pi/2)
RsS <- genR(rs, S = S)
mis.axis(RsS-S)
all.equal(mis.angle(RsS-S),abs(rs)) #TRUE

#If the central orientation is NOT id.SO3 then mis.angle(Rs) and abs(rs) are usual unequal
Rs <- genR(rs, S = genR(pi/8))
all.equal(mis.angle(Rs), abs(rs))  #Mean relative difference > 0
```

mis.axis *Misorientation axis*

Description

Determine the misorientation axis of a rotation.

Usage

```
mis.axis(x, ...)

## S3 method for class 'SO3'
mis.axis(x, ...)

## S3 method for class 'Q4'
mis.axis(x, ...)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
...	additional arguments.

Details

Every rotation can be interpreted as some reference coordinate system rotated about an axis through an angle. These quantities are referred to as the misorientation axis and misorientation angle, respectively, in the material sciences literature. This function returns the misorentation axis associated with a rotation assuming the reference coordinate system is the identity. The data must be centered before calling `mis.axis` if a different coordinate system is required.

Value

Axis in form of three dimensional vector of length one.

See Also

mis.angle

Examples

```
rs <- rcayley(20, kappa = 20)

#If the reference frame is set to id.SO3 then no centering is required
Rs <- genR(rs, S = id.SO3)
mis.axis(Rs)
all.equal(Rs, as.SO3(mis.axis(Rs), mis.angle(Rs)))
```

```
#For other reference frames, the data must be centered first
S <- genR(pi/2)
RsS <- genR(rs, S = S)
mis.axis(RsS-S)
all.equal(mis.angle(RsS-S),abs(rs)) #TRUE

Qs <- genR(rs, S = id.Q4, space = "Q4")
mis.axis(Qs)
all.equal(Qs, as.Q4(mis.axis(Qs), mis.angle(Qs)))
```

`Mises` *The circular-von Mises distribution*

Description

Density, distribution function and random generation for the circular-von Mises distribution with concentration `kappa` κ.

Usage

```
dvmises(r, kappa = 1, nu = NULL, Haar = TRUE)

pvmises(q, kappa = 1, nu = NULL, lower.tail = TRUE)

rvmises(n, kappa = 1, nu = NULL)
```

Arguments

`r`, `q`	vector of quantiles
`kappa`	concentration parameter.
`nu`	circular variance, can be used in place of `kappa`.
`Haar`	logical; if TRUE density is evaluated with respect to the Haar measure.
`lower.tail`	logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
`n`	number of observations. If `length(n)>1`, the length is taken to be the number required.

Details

The circular von Mises distribution with concentration κ has density

$$C_{\mathrm{M}}(r|\kappa) = \frac{1}{2\pi \mathrm{I}_0(\kappa)} e^{\kappa cos(r)}.$$

where $\mathrm{I}_0(\kappa)$ is the modified Bessel function of order 0.

Value

`dvmises`	gives the density
`pvmises`	gives the distribution function
`rvmises`	generates random deviates

See Also

Angular-distributions for other distributions in the rotations package.

Examples

```
r <- seq(-pi, pi, length = 500)

#Visualize the von Mises density fucntion with respect to the Haar measure
plot(r, dvmises(r, kappa = 10), type = "l", ylab = "f(r)", ylim = c(0, 100))

#Visualize the von Mises density fucntion with respect to the Lebesgue measure
plot(r, dvmises(r, kappa = 10, Haar = FALSE), type = "l", ylab = "f(r)")

#Plot the von Mises CDF
plot(r,pvmises(r,kappa = 10), type = "l", ylab = "F(r)")

#Generate random observations from von Mises distribution
rs <- rvmises(20, kappa = 1)
hist(rs, breaks = 10)
```

---

nickel *Nickel electron backscatter diffraction data set*

Description

This data set consists of electron backscatter diffraction (EBSD) data obtained by scanning a fixed 12.5 μm-by-10 μm nickel surface at individual locations spaced 0.2 μm apart. This scan was repeated 14 times for each of the 3,449 locations yielding a total of 48,286 observations. Every observation corresponds to the orientation, expressed as a rotation matrix, of a cubic crystal on the metal surface at a particular location. Be aware that there are missing values and erroneous scans at some locations and scans. See Bingham et al. (2009) and Bingham et al. (2010) for more details and analysis.

Usage

```
nickel
```

Format

A data frame with 48,286 rows and the following 13 columns:

`xpos` location x position

`ypos` location y position

`location` Location number for easy reference

`rep` Replicate scan identifier

`V1` First element of x-axis describing crystal orientation at corresponding location

`V2` Second element of x-axis describing crystal orientation at corresponding location

`V3` Third element of x-axis describing crystal orientation at corresponding location

`V4` First element of y-axis describing crystal orientation at corresponding location

`V5` Second element of y-axis describing crystal orientation at corresponding location

`V6` Third element of y-axis describing crystal orientation at corresponding location

`V7` First element of z-axis describing crystal orientation at corresponding location

`V8` Second element of z-axis describing crystal orientation at corresponding location

`V9` Third element of z-axis describing crystal orientation at corresponding location

Source

The data set was collected by the Ames Lab located in Ames, IA.

References

1. Bingham, M. A., Nordman, D., & Vardeman, S. (2009). "Modeling and inference for measured crystal orientations and a tractable class of symmetric distributions for rotations in three dimensions." Journal of the American Statistical Association, 104(488), pp. 1385-1397.
2. Bingham, M. A., Lograsso, B. K., & Laabs, F. C. (2010). "A statistical analysis of the variation in measured crystal orientations obtained through electron backscatter diffraction." Ultramicroscopy, 110(10), pp. 1312-1319.
3. Stanfill, B., Genschel, U., & Heike, H. (2013). "Point estimation of the central orientation of random rotations". Technometrics, 55(4), pp. 524-535.

Examples

```
# Subset the data to include only the first scan
Rep1 <- subset(nickel, rep == 1)

# Get a rough idea of how the grain map looks by plotting the first
# element of the rotation matrix at each location
ggplot2::qplot(xpos, ypos, data = Rep1, colour = V1, size = I(2))

# Focus in on a particular location, for example location 698
Rs <- subset(nickel, location == 698)

# Translate the Rs data.frame into an object of class 'SO3'
Rs <- as.SO3(Rs[,5:13])
```

```
# Some observations are not rotations, remove them
Rs <- Rs[is.SO3(Rs),]

# Estimate the central orientation with the average
mean(Rs)

# Re-estimate central orientation robustly
median(Rs)


  # Visualize the location, there appears to be two groups
  plot(Rs, col = c(1, 2, 3))
```

---

plot *Visualizing random rotations*

Description

This function produces a static three-dimensional globe onto which one of the columns of the provided sample of rotations is projected. The data are centered around a user-specified rotation matrix. The static plot uses ggplot2. Interactive plots are no longer supported.

Usage

```
## S3 method for class 'SO3'
plot(
  x,
  center = mean(x),
  col = 1,
  to_range = FALSE,
  show_estimates = NULL,
  label_points = NULL,
  mean_regions = NULL,
  median_regions = NULL,
  alp = NULL,
  m = 300,
  interactive = FALSE,
  ...
)

## S3 method for class 'Q4'
plot(
  x,
  center = mean(x),
  col = 1,
  to_range = FALSE,
```

```
  show_estimates = NULL,
  label_points = NULL,
  mean_regions = NULL,
  median_regions = NULL,
  alp = NULL,
  m = 300,
  interactive = FALSE,
  ...
)
```

Arguments

x	n rotations in SO3 or Q4 format.
center	rotation about which to center the observations.
col	integer or vector comprised of 1, 2, 3 indicating which column(s) to display. If length(col)>1 then each eyeball is labelled with the corresponding axis.
to_range	logical; if TRUE only part of the globe relevant to the data is displayed
show_estimates	character vector to specify which of the four estimates of the principal direction to show. Possibilities are "all", "proj.mean", "proj.median", "geom.mean", "geom.median".
label_points	vector of labels.
mean_regions	character vector to specify which of the three confidence regions to show for the projected mean. Possibilities are "all", "trans.theory","trans.bootstrap, "direct.theory", "direct.bootstrap".
median_regions	character vector to specify which of the three confidence regions to show for the projected median. Possibilities are "all", "theory", "bootstrap."
alp	alpha level to be used for confidence regions. See region for more details.
m	number of bootstrap replicates to use in bootstrap confidence regions.
interactive	deprecated; sphereplot was set to be removed from CRAN and was going to take this package down with it
...	parameters passed onto the points layer.

Value

A visualization of rotation data.

Examples

```
r <- rvmises(200, kappa = 1.0)
Rs <- genR(r)
plot(Rs, center = mean(Rs), show_estimates = "proj.mean", shape = 4)


  # Z is computed internally and contains information on depth
  plot(
    Rs,
```

```
    center = mean(Rs),
    show_estimates = c("proj.mean", "geom.mean"),
    label_points = sample(LETTERS, 200, replace = TRUE)
) +
  ggplot2::aes(size = Z, alpha = Z) +
  ggplot2::scale_size(limits = c(-1, 1), range = c(0.5, 2.5))
```

---

pointsXYZ *Project rotation data onto sphere*

Description

Projection of rotation matrices onto sphere with given center.

Usage

```
pointsXYZ(data, center = id.SO3, column = 1)
```

Arguments

data	data frame of rotation matrices in 3×3 matrix representation.
center	rotation matrix about which to center the observations.
column	integer 1 to 3 indicating which column to display.

Value

Data frame with columns X, Y, Z standing for the respective coordinates in 3D space.

Examples

```
Rs<-ruars(20, rcayley)

#Project the sample's 3 axes onto the 3-shere centered at the identity rotation

pointsXYZ(Rs, center = id.SO3, column = 1)  #x-axis
pointsXYZ(Rs, center = id.SO3, column = 2)  #y-axis
pointsXYZ(Rs, center = id.SO3, column = 3)  #z-axis
```

prentice *Transformation based asymptotic confidence region*

Description

Find the radius of a $100(1-\alpha)\%$ confidence region for the projected mean based on a result from directional statistics.

Usage

```
prentice(x, alp)

## S3 method for class 'Q4'
prentice(x, alp = NULL)

## S3 method for class 'SO3'
prentice(x, alp = NULL)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
alp	alpha level desired, e.g. 0.05 or 0.10.

Details

Compute the radius of a $100(1-\alpha)\%$ confidence region for the central orientation based on the projected mean estimator using the method due to *prentice1986*. For a rotation specific version see *rancourt2000*. The variability in each axis is different so each axis will have its own radius.

prentice1986, rancourt2000

Value

Radius of the confidence region centered at the projected mean for each of the x-, y- and z-axes.

See Also

bayesCR, fisheretal, chang, zhang

Examples

```
Qs<-ruars(20, rcayley, kappa = 100, space = 'Q4')

# The prentice method can be accessed from the "region" function or the "prentice" function
region(Qs, method = "transformation", type = "asymptotic", alp = 0.1, estimator = "mean")
prentice(Qs, alp = 0.1)
```

project.SO3 *Projection into SO(3)*

Description

Project an arbitrary 3×3 matrix into $SO(3)$.

Usage

```
project.SO3(M)
```

Arguments

M	3×3 matrix to project into $SO(3)$.

Details

This function uses the process detailed in Section 3.1 of *moakher02* to project an arbitrary 3×3 matrix into $SO(3)$. More specifically it finds the closest orthogonal 3-by-3 matrix with determinant one to the provided matrix.

Value

Projection of $\boldsymbol{M}$ into $SO(3)$.

See Also

mean.SO3, median.SO3

Examples

```
#Project an arbitrary 3x3 matrix into SO(3)
M<-matrix(rnorm(9), 3, 3)
project.SO3(M)

#Project a sample arithmetic mean into SO(3), same as 'mean'
Rs <- ruars(20, rcayley)
Rbar <- colSums(Rs)/nrow(Rs)
project.SO3(Rbar)             #The following is equivalent
mean(Rs)
```

Q4 *'Q4' class for storing rotation data as quaternions*

Description

Creates or tests for objects of class "Q4".

Usage

```
as.Q4(x, ...)

## Default S3 method:
as.Q4(x, theta = NULL, ...)

## S3 method for class 'SO3'
as.Q4(x, ...)

## S3 method for class 'Q4'
as.Q4(x, ...)

## S3 method for class 'data.frame'
as.Q4(x, ...)

is.Q4(x)

id.Q4
```

Arguments

x	object to be coerced or tested
...	additional arguments.
theta	vector or single rotation angle; if `length(theta)==1`, the same theta is used for all axes

Format

`id.Q4` is the identity rotation given by the matrix $[1, 0, 0, 0]^\top$.

Details

Construct a single or sample of rotations in 3-dimensions in quaternion form. Several possible inputs for `x` are possible and they are differentiated based on their class and dimension.

For `x` an n-by-3 matrix or a vector of length 3, the angle-axis representation of rotations is utilized. More specifically, each quaternion can be interpreted as a rotation of some reference frame about the axis U (of unit length) through the angle θ. For each axis and angle the quaternion is formed through

$$q = [cos(\theta/2), sin(\theta/2)U]^\top.$$

The object x is treated as if it has rows U and theta is a vector or angles. If no angle is supplied then the length of each axis is taken to be the angle of rotation theta.

For x an n-by-9 matrix of rotation matrices or an object of class "SO3", this function will return the quaternion equivalent of x. See SO3 or the vignette "rotations-intro" for more details on rotation matrices.

For x an n-by-4 matrix, rows are treated as quaternions; rows that aren't of unit length are made unit length while the rest are returned untouched. A message is printed if any of the rows are not quaternions.

For x a "data.frame", it is translated into a matrix of the same dimension and the dimensionality of x is used to determine the data type: angle-axis, quaternion or rotation (see above). As demonstrated below, is.Q4 may return TRUE for a data frame, but the functions defined for objects of class 'Q4' will not be called until as.Q4 has been used.

Value

as.Q4	coerces its object into a Q4 type
is.Q4	returns TRUE or FALSE depending on whether its argument satisfies the conditions to be an quaternion; namely it must be four-dimensional and of unit length

Examples

```
# Pull off subject 1's wrist measurements
Subj1Wrist <- subset(drill, Subject == '1' & Joint == 'Wrist')

                                ## The measurements are in columns 5:8
all(is.Q4(Subj1Wrist[,5:8]))   #TRUE, even though Qs is a data.frame, the rows satisfy the
                                #conditions necessary to be quaternions BUT,
                                #S3 methods (e.g. 'mean' or 'plot') for objects of class
                                #'Q4' will not work until 'as.Q4' is used

Qs <- as.Q4(Subj1Wrist[,5:8])  #Coerce measurements into 'Q4' type using as.Q4.data.frame
mean(Qs)                 #Estimate central orientation for subject 1's wrist, see ?mean.Q4
Rs <- as.SO3(Qs)            #Coerce a 'Q4' object into rotation matrix format, see ?as.SO3

#Visualize the measurements, see ?plot.Q4 for more

  plot(Qs, col = c(1, 2, 3))
```

---

region *Confidence and credible regions for the central orientation*

Description

Find the radius of a $100(1-\alpha)\%$ confidence or credible region for the central orientation based on the projected mean or median. For more on the currently available methods see prentice, fisheretal, chang, zhang and bayesCR.

Usage

```
region(x, method, type, estimator, alp = NULL, ...)

## S3 method for class 'Q4'
region(x, method, type, estimator, alp = NULL, ...)

## S3 method for class 'SO3'
region(x, method, type, estimator, alp = NULL, ...)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
method	character string specifying which type of interval to report, "bayes", "transformation" or "direct" based theory.
type	character string, "bootstrap" or "asymptotic" are available. For Bayes regions, give the type of likelihood: "Cayley","Mises" or "Fisher."
estimator	character string either "mean" or "median." Note that not all method/type combinations are available for both estimators.
alp	the alpha level desired, e.g. 0.05 or 0.10.
...	additional arguments that are method specific.

Value

For frequentist regions only the radius of the confidence region centered at the specified estimator is returned. For Bayes regions the posterior mode and radius of the credible region centered at that mode is returned.

See Also

bayesCR, prentice, fisheretal, chang, zhang

Examples

```
Rs <- ruars(20, rvmises, kappa = 10)

# Compare the region sizes that are currently available

region(Rs, method = "transformation", type = "asymptotic", estimator = "mean", alp = 0.1)
region(Rs, method = "transformation", type = "bootstrap", estimator = "mean",
alp = 0.1, symm = TRUE)
region(Rs, method = "direct", type = "bootstrap", estimator = "mean", alp = 0.1, m = 100)
region(Rs, method = "direct", type = "asymptotic", estimator = "mean", alp = 0.1)

region(Rs, method = "Bayes", type = "Mises", estimator = "mean",
     S0 = mean(Rs), kappa0 = 10, tuneS = 5000, tuneK = 1, burn_in = 1000, alp = .01, m = 5000)
```

`rot.dist` *Rotational distance*

Description

Calculate the extrinsic or intrinsic distance between two rotations.

Usage

```
rot.dist(x, ...)

## S3 method for class 'SO3'
rot.dist(x, R2 = id.SO3, method = "extrinsic", p = 1, ...)

## S3 method for class 'Q4'
rot.dist(x, Q2 = id.Q4, method = "extrinsic", p = 1, ...)
```

Arguments

`x`	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
`...`	additional arguments.
`R2, Q2`	a single, second rotation in the same parametrization as x.
`method`	string indicating "extrinsic" or "intrinsic" method of distance.
`p`	the order of the distance.

Details

This function will calculate the intrinsic (Riemannian) or extrinsic (Euclidean) distance between two rotations. `R2` and `Q2` are set to the identity rotations by default. For rotations R_1 and R_2 both in $SO(3)$, the Euclidean distance between them is

$$||R_1 - R_2||_F$$

where $|| \cdot ||_F$ is the Frobenius norm. The Riemannian distance is defined as

$$||Log(R_1^\top R_2)||_F$$

where Log is the matrix logarithm, and it corresponds to the misorientation angle of $R_1^\top R_2$. See the vignette 'rotations-intro' for a comparison of these two distance measures.

Value

The rotational distance between each rotation in x and R2 or Q2.

Examples

```
rs <- rcayley(20, kappa = 10)
Rs <- genR(rs, S = id.SO3)
dEs <- rot.dist(Rs,id.SO3)
dRs <- rot.dist(Rs, id.SO3 , method = "intrinsic")

#The intrinsic distance between the true central orientation and each observation
#is the same as the absolute value of observations' respective misorientation angles
all.equal(dRs, abs(rs))              #TRUE

#The extrinsic distance is related to the intrinsic distance
all.equal(dEs, 2*sqrt(2)*sin(dRs/2)) #TRUE
```

`rotations` *A package for working with rotation data.*

Description

This package implements tools for working with rotational data: it allows simulation from the most commonly used distributions on $SO(3)$, it includes methods for different mean and median type estimators for the central orientation of a sample, it provides confidence regions for those estimates and it includes a novel visualization technique for rotation data.

Author(s)

Maintainer: Bryan Stanfill <bstanfill2003@gmail.com>

Authors:

- Bryan Stanfill <bstanfill2003@gmail.com>
- Heike Hofmann
- Ulrike Genschel

Other contributors:

- Aymeric Stamm <aymeric.stamm@math.cnrs.fr> (ORCID) [contributor]
- Luciano Selzer [contributor]

See Also

Useful links:

- https://github.com/stanfill/rotationsC
- Report bugs at https://github.com/stanfill/rotationsC/issues

rotdist.sum *Sample distance*

Description

Compute the sum of the p^{th} order distances between each row of x and S.

Usage

```
rotdist.sum(x, S = genR(0, space = class(x)), method = "extrinsic", p = 1)

## S3 method for class 'SO3'
rotdist.sum(x, S = id.SO3, method = "extrinsic", p = 1)

## S3 method for class 'Q4'
rotdist.sum(x, S = id.Q4, method = "extrinsic", p = 1)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
S	the individual matrix of interest, usually an estimate of the mean.
method	type of distance used method in "extrinsic" or "intrinsic"
p	the order of the distances to compute.

Value

The sum of the pth order distance between each row of x and S.

See Also

rot.dist

Examples

```
Rs <- ruars(20, rvmises, kappa = 10)

SE1 <- median(Rs)                     #Projected median
SE2 <- mean(Rs)                       #Projected mean
SR2 <- mean(Rs, type = "geometric")   #Geometric mean

#I will use "rotdist.sum" to verify these three estimators minimize the
#loss function they are designed to minimize relative to the other esimators.
#All of the following statements should evaluate to "TRUE"

#The projected mean minimizes the sum of squared Euclidean distances
rotdist.sum(Rs, S = SE2, p = 2) < rotdist.sum(Rs, S = SE1, p = 2)
rotdist.sum(Rs, S = SE2, p = 2) < rotdist.sum(Rs, S = SR2, p = 2)
```

```
#The projected median minimizes the sum of first order Euclidean distances
rotdist.sum(Rs, S = SE1, p = 1) < rotdist.sum(Rs, S = SE2, p = 1)
rotdist.sum(Rs, S = SE1, p = 1) < rotdist.sum(Rs, S = SR2, p = 1)

#The geometric mean minimizes the sum of squared Riemannian distances
rotdist.sum(Rs, S = SR2, p = 2, method = "intrinsic") <
                 rotdist.sum(Rs, S = SE1, p = 2, method = "intrinsic")
rotdist.sum(Rs, S = SR2, p = 2, method = "intrinsic") <
                 rotdist.sum(Rs, S = SE2, p = 2, method = "intrinsic")
```

---

skew.exp *Matrix exponential*

Description

Compute the matrix exponential for skew-symmetric matrices according to the usual Taylor expansion. The expansion is significantly simplified for skew-symmetric matrices, see *moakher02*. Maps a matrix belonging to the lie algebra $so(3)$ into the lie group $SO(3)$.

Usage

```
skew.exp(x)
```

Arguments

x	single 3×3 skew-symmetric matrix or $n\times9$ sample of skew-symmetric matrices.

Details

moakher02

Value

Matrix $e^{\boldsymbol{H}}$ in $SO(3)$.

Examples

```
Rs <- ruars(20, rcayley)
lRs <- log(Rs)           #Take the matrix logarithm for rotation matrices
Rs2 <- skew.exp(lRs)     #Go back to rotation matrices
all.equal(Rs, Rs2)
```

SO3 *'SO3' class for storing rotation data as rotation matrices*

Description

Creates or tests for objects of class "SO3".

Usage

```
as.SO3(x, ...)

## Default S3 method:
as.SO3(x, theta = NULL, ...)

## S3 method for class 'Q4'
as.SO3(x, ...)

## S3 method for class 'SO3'
as.SO3(x, ...)

## S3 method for class 'data.frame'
as.SO3(x, ...)

is.SO3(x)

id.SO3
```

Arguments

x	object to be coerced or tested; see details for possible forms
...	additional arguments.
theta	vector or single rotation angle; if `length(theta)==1` the same theta is used for all axes

Format

`id.SO3` is the identity rotation given by the the 3-by-3 identity matrix.

Details

Construct a single or sample of rotations in 3-dimensions in 3-by-3 matrix form. Several possible inputs for `x` are possible and they are differentiated based on their class and dimension.

For `x` an n-by-3 matrix or a vector of length 3, the angle-axis representation of rotations is utilized. More specifically, each rotation matrix can be interpreted as a rotation of some reference frame

about the axis U (of unit length) through the angle θ. If a single axis (in matrix or vector format) or matrix of axes are provided for x, then for each axis and angle the matrix is formed through

$$R = \exp[\Phi(U\theta)]$$

where U is replace by x. If axes are provided but theta is not provided then the length of each axis is taken to be the angle of rotation, theta.

For x an n-by-4 matrix of quaternions or an object of class "Q4", this function will return the rotation matrix equivalent of x. See Q4 or the vignette "rotations-intro" for more details on quaternions.

For x an n-by-9 matrix, rows are treated as 3-by-3 matrices; rows that don't form matrices in SO(3) are projected into SO(3) and those that are already in SO(3) are returned untouched. See project.SO3 for more on projecting arbitrary matrices into SO(3). A message is printed if any of the rows are not proper rotations.

For x a "data.frame", it is translated into a matrix of the same dimension and the dimensionality of x is used to determine the data type: angle-axis, quaternion or rotation. As demonstrated below, is.SO3 may return TRUE for a data frame, but the functions defined for objects of class "SO3" will not be called until as.SO3 has been used.

Value

as.SO3	coerces provided data into an SO3 type.
is.SO3	returns TRUE or False depending on whether its argument satisfies the conditions to be an rotation matrix. Namely, has determinant one and its transpose is its inverse.

Examples

```
# Select one location to focus on
Loc698 <- subset(nickel, location == 698)

is.SO3(Loc698[,5:13])       #Some of the rows are not rotations due to rounding or entry errors
                               #as.SO3 will project matrices not in SO(3) to SO(3)

Rs <- as.SO3(Loc698[,5:13])    #Translate the Rs data.frame into an object of class 'SO3'
                          #Rows 4, 6 and 13 are not in SO(3) so they are projected to SO(3)

mean(Rs)                       #Estimate the central orientation with the average
median(Rs)                     #Re-estimate central orientation robustly
Qs <- as.Q4(Rs)                #Coerse into "SO3" format, see ?as.SO3 for more

#Visualize the location, there appears to be two groups

  plot(Rs, col = c(1, 2, 3))
```

`tail` *Return the First or Last Parts of an Object*

Description

Returns the first or last parts of a vector, matrix, array, table, data frame or function. Since `head()` and `tail()` are generic functions, they have been extended to other classes, including `"ts"` from **stats**.

Usage

```
## S3 method for class 'SO3'
tail(x, n = 6L, addrownums = TRUE, ...)

## S3 method for class 'Q4'
tail(x, n = 6L, addrownums = TRUE, ...)
```

Arguments

`x`	an object
`n`	an integer vector of length up to `dim(x)` (or 1, for non-dimensioned objects). A `logical` is silently coerced to integer. Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of `n[i]` includes the first/last `n[i]` indices in that dimension, while a negative value excludes the last/first `abs(n[i])`, including all remaining indices. `NA` or non-specified values (when `length(n) < length(dim(x))`) select all indices in that dimension. Must contain at least one non-missing value.
`addrownums`	deprecated - `keepnums` should be used instead. Taken as the value of `keepnums` if it is explicitly set when `keepnums` is not.
`...`	arguments to be passed to or from other methods.

Details

For vector/array based objects, `head()` (`tail()`) returns a subset of the same dimensionality as `x`, usually of the same class. For historical reasons, by default they select the first (last) 6 indices in the first dimension ("rows") or along the length of a non-dimensioned vector, and the full extent (all indices) in any remaining dimensions. `head.matrix()` and `tail.matrix()` are exported.

The default and array(/matrix) methods for `head()` and `tail()` are quite general. They will work as is for any class which has a `dim()` method, a `length()` method (only required if `dim()` returns `NULL`), and a `[` method (that accepts the `drop` argument and can subset in all dimensions in the dimensioned case).

For functions, the lines of the deparsed function are returned as character strings.

When `x` is an array(/matrix) of dimensionality two and more, `tail()` will add dimnames similar to how they would appear in a full printing of `x` for all dimensions `k` where `n[k]` is specified and non-missing and `dimnames(x)[[k]]` (or `dimnames(x)` itself) is `NULL`. Specifically, the form of the added dimnames will vary for different dimensions as follows:

k=1 **(rows):** "[n,]" (right justified with whitespace padding)

k=2 **(columns):** "[,n]" (with *no* whitespace padding)

k>2 **(higher dims):** "n", i.e., the indices as *character* values

Setting keepnums = FALSE suppresses this behaviour.

As data.frame subsetting ('indexing') keeps attributes, so do the head() and tail() methods for data frames.

The auxiliary function .checkHT(d, n) is useful in head(x, n) or tail(x, n) methods, checking validity of d <- dim(x) and n.

Value

An object (usually) like x but generally smaller. Hence, for arrays, the result corresponds to x[.., drop=FALSE]. For ftable objects x, a transformed format(x).

Note

For array inputs the output of tail when keepnums is TRUE, any dimnames vectors added for dimensions >2 are the original numeric indices in that dimension *as character vectors*. This means that, e.g., for 3-dimensional array arr, tail(arr, c(2,2,-1))[, , 2] and tail(arr, c(2,2,-1))[, , "2"] may both be valid but have completely different meanings.

Author(s)

Patrick Burns, improved and corrected by R-Core. Negative argument added by Vincent Goulet. Multi-dimension support added by Gabriel Becker.

Examples

```
head(letters)
head(letters, n = -6L)

head(freeny.x, n = 10L)
head(freeny.y)

head(gait) # 3d array
head(gait, c(6L, 2L))
head(gait, c(6L, 2L, -1L))

tail(letters)
tail(letters, n = -6L)

tail(freeny.x)
## the bottom-right "corner" :
tail(freeny.x, n = c(4, 2))
tail(freeny.y)

tail(gait)
tail(gait, c(6L, 2L))
tail(gait, c(6L, 2L, -1L))
```

```
## gait without dimnames --> keepnums showing original row/col numbers
a3 <- gait ; dimnames(a3) <- NULL
tail(a3, c(6, 2, -1))# keepnums = TRUE is default here!
tail(a3, c(6, 2, -1),  keepnums = FALSE)

## data frame w/ a (non-standard) attribute:
treeS <- structure(trees, foo = "bar")
(n <- nrow(treeS))
stopifnot(exprs = { # attribute is kept
    identical(htS <- head(treeS), treeS[1:6, ])
    identical(attr(htS, "foo") , "bar")
    identical(tlS <- tail(treeS), treeS[(n-5):n, ])
    ## BUT if I use "useAttrib(.)", this is *not* ok, when n is of length 2:
    ## --- because [i,j]-indexing of data frames *also* drops "other" attributes ..
    identical(tail(treeS, 3:2), treeS[(n-2):n, 2:3] )
})

tail(library) # last lines of function

head(stats::ftable(Titanic))

## 1d-array (with named dim) :
a1 <- array(1:7, 7); names(dim(a1)) <- "O2"
stopifnot(exprs = {
  identical( tail(a1, 10), a1)
  identical( head(a1, 10), a1)
  identical( head(a1, 1), a1 [1 , drop=FALSE] ) # was a1[1] in R <= 3.6.x
  identical( tail(a1, 2), a1[6:7])
  identical( tail(a1, 1), a1 [7 , drop=FALSE] ) # was a1[7] in R <= 3.6.x
})
```

UARS *Generic UARS Distribution*

Description

Density, distribution function and random generation for the the generic uniform axis-random spin (UARS) class of distributions.

Usage

```
duars(R, dangle, S = id.SO3, kappa = 1, ...)

puars(R, pangle = NULL, S = id.SO3, kappa = 1, ...)

ruars(n, rangle, S = NULL, kappa = 1, space = "SO3", ...)
```

Arguments

R	Value at which to evaluate the UARS density.
dangle	The function to evaluate the angles from, e.g. dcayley, dvmises, dfisher, dhaar.
S	central orientation of the distribution.
kappa	concentration parameter.
...	additional arguments.
pangle	The form of the angular density, e.g. pcayley, pvmises, pfisher, phaar.
n	number of observations. If length(n)>1, the length is taken to be the number required.
rangle	The function from which to simulate angles, e.g. rcayley, rvmises, rhaar, rfisher.
space	indicates the desired representation: matrix ("SO3") or quaternion ("Q4").

Details

For the rotation R with central orientation S and concentration κ the UARS density is given by

$$f(R|S,\kappa) = \frac{4\pi}{3 - tr(S^\top R)} C(\cos^{-1}[tr(S^\top R) - 1]/2|\kappa)$$

where $C(r|\kappa)$ is one of the Angular-distributions.

bingham09

Value

duars	gives the density
puars	gives the distribution function. If pangle is left empty, the empirical CDF is returned.
ruars	generates random deviates

See Also

For more on the angular distribution options see Angular-distributions.

Examples

```
#Generate random rotations from the Cayley-UARS distribution with central orientation
#rotated about the y-axis through pi/2 radians
S <- as.SO3(c(0, 1, 0), pi/2)
Rs <- ruars(20, rangle = rcayley, kappa = 1, S = S)

rs <- mis.angle(Rs-S)                          #Find the associated misorientation angles
frs <- duars(Rs, dcayley, kappa = 10, S = S)  #Compute UARS density evaluated at each rotations
plot(rs, frs)

cdf <- puars(Rs, pcayley, S = S)          #By supplying 'pcayley', it is used to compute the
plot(rs, cdf)                                    #the CDF

ecdf <- puars(Rs, S = S)                 #No 'puars' arguement is supplied so the empirical
plot(rs, ecdf)                                   #cdf is returned
```

vmises.kappa *Circular variance and concentration parameter*

Description

Return the concentration parameter that corresponds to a given circular variance.

Usage

```
vmises.kappa(nu)
```

Arguments

nu	circular variance

Details

The concentration parameter κ does not translate across circular distributions. A commonly used measure of spread in circular distributions that does translate is the circular variance defined as $\nu = 1 - E[\cos(r)]$ where $E[\cos(r)]$ is the mean resultant length. See *mardia2000* for more details. This function translates the circular variance ν into the corresponding concentration parameter κ for the circular-von Mises distribution. For numerical stability, a maximum κ of 500 is returned.

mardia2000

Value

Concentration parameter corresponding to nu.

See Also

Mises

Examples

```
# Find the concentration parameter for circular variances 0.25, 0.5, 0.75
vmises.kappa(0.25)
vmises.kappa(0.5)
vmises.kappa(0.75)
```

`weighted.mean` *Weighted mean rotation*

Description

Compute the weighted geometric or projected mean of a sample of rotations.

Usage

```
## S3 method for class 'SO3'
weighted.mean(
  x,
  w = NULL,
  type = "projected",
  epsilon = 1e-05,
  maxIter = 2000,
  ...
)

## S3 method for class 'Q4'
weighted.mean(
  x,
  w = NULL,
  type = "projected",
  epsilon = 1e-05,
  maxIter = 2000,
  ...
)
```

Arguments

`x`	$n \times p$ matrix where each row corresponds to a random rotation in matrix form ($p = 9$) or quaternion ($p = 4$) form.
`w`	vector of weights the same length as the number of rows in x giving the weights to use for elements of x. Default is `NULL`, which falls back to the usual `mean` function.
`type`	string indicating "projected" or "geometric" type mean estimator.
`epsilon`	stopping rule for the geometric method.
`maxIter`	maximum number of iterations allowed before returning most recent estimate.
`...`	only used for consistency with mean.default.

Details

This function takes a sample of 3D rotations (in matrix or quaternion form) and returns the weighted projected arithmetic mean $\widehat{\boldsymbol{S}}_P$ or geometric mean $\widehat{\boldsymbol{S}}_G$ according to the `type` option. For a sample

of n rotations in matrix form $\boldsymbol{R}_i \in SO(3), i = 1, 2, \ldots, n$, the weighted mean is defined as

$$\widehat{\boldsymbol{S}} = argmin_{\boldsymbol{S} \in SO(3)} \sum_{i=1}^{n} w_i d^2(\boldsymbol{R}_i, \boldsymbol{S})$$

where d is the Riemannian or Euclidean distance. For more on the projected mean see *moakher02* and for the geometric mean see *manton04*.

moakher02

Value

Weighted mean of the sample in the same parametrization.

See Also

median.SO3, mean.SO3, bayes.mean

Examples

```
Rs <- ruars(20, rvmises, kappa = 0.01)

# Find the equal-weight projected mean
mean(Rs)

# Use the rotation misorientation angle as weight
wt <- abs(1 / mis.angle(Rs))
weighted.mean(Rs, wt)

rot.dist(mean(Rs))

# Usually much smaller than unweighted mean
rot.dist(weighted.mean(Rs, wt))

# Can do the same thing with quaternions
Qs <- as.Q4(Rs)
mean(Qs)
wt <- abs(1 / mis.angle(Qs))
weighted.mean(Qs, wt)
rot.dist(mean(Qs))
rot.dist(weighted.mean(Qs, wt))
```

---

zhang *M-estimator theory pivotal bootstrap confidence region*

Description

Compute the radius of a $100(1 - \alpha)\%$ confidence region for the central orientation based on M-estimation theory.

Usage

```
zhang(x, estimator, alp = NULL, m = 300)

## S3 method for class 'SO3'
zhang(x, estimator, alp = NULL, m = 300)

## S3 method for class 'Q4'
zhang(x, estimator, alp = NULL, m = 300)
```

Arguments

x	$n \times p$ matrix where each row corresponds to a random rotation in matrix ($p = 9$) or quaternion ($p = 4$) form.
estimator	character string either "mean" or "median."
alp	alpha level desired, e.g. 0.05 or 0.10.
m	number of replicates to use to estimate the critical value.

Details

Compute the radius of a $100(1 - \alpha)\%$ confidence region for the central orientation based on the projected mean estimator using the method due to Zhang & Nordman (2009) (unpublished MS thesis). By construction each axis will have the same radius so the radius reported is for all three axis. A normal theory version of this procedure uses the theoretical chi-square limiting distribution and is given by the chang option. This method is called "direct" because it used M-estimation theory for SO(3) directly instead of relying on transforming a result from directional statistics as prentice and fisheretal do.

Value

Radius of the confidence region centered at the specified estimator.

See Also

bayesCR, prentice, fisheretal, chang

Examples

```
Rs <- ruars(20, rcayley, kappa = 100)

# The zhang method can be accesed from the "region" function or the "zhang" function
# They will be different because it is a bootstrap.
region(Rs, method = "direct", type = "bootstrap", alp = 0.1, estimator = "mean")
zhang(Rs, estimator = "mean", alp = 0.1)
```

Index

∗ **datasets**
drill, 13
nickel, 35
+.Q4 *(*Arithmetic*)*, 4
+.SO3 *(*Arithmetic*)*, 4
-.Q4 *(*Arithmetic*)*, 4
-.SO3 *(*Arithmetic*)*, 4

Angular-distributions, 3, *9*, *15*, *21*, *25*, *35*, *54*
Arithmetic, 4
array, *22*, *52*
as.Q4 *(*Q4*)*, 42
as.SO3 *(*SO3*)*, 49
attributes, *22*, *52*

bayes.mean, 5, *29*, *31*, *57*
bayesCR, 7, *12*, *17*, *40*, *43, 44*, *58*

Cayley, *3*, *5*, *7*, 8, *10*, *15*, *27*
cayley.kappa, 9
center, 10
chang, *8*, 11, *17*, *40*, *43, 44*, *58*

data.frame, *22*, *52*
dcayley *(*Cayley*)*, 8
dfisher *(*Fisher*)*, 14
dhaar *(*Haar*)*, 20
discord, 12
dmaxwell *(*Maxwell*)*, 25
drill, 13
duars *(*UARS*)*, 53
dvmises *(*Mises*)*, 34

Fisher, *3*, *5*, *7*, 14, *16*, *27*
fisher.kappa, 16
fisheretal, *8*, *12*, 17, *40*, *43, 44*, *58*
ftable, *22*, *52*

genR, 18
gradient.search, 19

Haar, *3*, 20
head, 21

id.Q4 *(*Q4*)*, 42
id.SO3 *(*SO3*)*, 49
is.Q4 *(*Q4*)*, 42
is.SO3 *(*SO3*)*, 49

log.SO3, 24

Maxwell, *3*, 25, *26*
maxwell.kappa, 26
MCMCSO3, 27
mean, 28
mean.SO3, *6*, *31*, *41*, *57*
median, 30
median.SO3, *6*, *29*, *41*, *57*
mis.angle, 31, *33*
mis.axis, *32*, 33
Mises, *3*, *5*, *7*, *27*, 34, *55*

nickel, 35

pcayley *(*Cayley*)*, 8
pfisher *(*Fisher*)*, 14
phaar *(*Haar*)*, 20
plot, 37
pmaxwell *(*Maxwell*)*, 25
pointsXYZ, 39
prentice, *8*, *12*, *17*, 40, *43, 44*, *58*
project.SO3, 41, *50*
puars *(*UARS*)*, 53
pvmises *(*Mises*)*, 34

Q4, 42, *50*

rcayley *(*Cayley*)*, 8
region, *38*, 43
rfisher *(*Fisher*)*, 14
rhaar *(*Haar*)*, 20
rmaxwell *(*Maxwell*)*, 25

rot.dist, 45, *47*
rotations, 46
rotations-package *(*rotations*)*, 46
rotdist.sum, 47
ruars *(*UARS*)*, 53
rvmises *(*Mises*)*, 34

skew.exp, 48
SO3, *43*, 49

tail, 51
ts, *21*, *51*

UARS, *20*, 53

vmises.kappa, 55

weighted.mean, 56
weighted.mean.SO3, *29*, *31*

zhang, *8*, *12*, *17*, *40*, *43*, *44*, 57