Package 'rotasym'

November 12, 2026

Type Package

Title Tests for Rotational Symmetry on the Hypersphere

Version 1.4.0

Date 2026-11-11

Description Implementation of the tests for rotational symmetry on the hypersphere proposed in García-Portugués, Paindaveine and Verdebout (2020) <doi:10.1080/01621459.2019.1665527>. The package also implements the proposed distributions on the hypersphere, based on the tangent-normal decomposition, and allows for the replication of the data application considered in the paper.

License GPL-3

LazyData true

Depends R (>= 3.5.0), Rcpp

Imports DirStats (>= 1.0.0)

Suggests dplyr, scatterplot3d, testthat (>= 3.0.0), viridisLite, withr

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/egarpor/rotasym

BugReports https://github.com/egarpor/rotasym/issues

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Eduardo García-Portugués [aut, cre] (ORCID: <https://orcid.org/0000-0002-9224-4111>), Davy Paindaveine [aut] (ORCID: <https://orcid.org/0000-0002-6758-4261>), Thomas Verdebout [aut] (ORCID: <https://orcid.org/0000-0002-3277-9587>)

Maintainer Eduardo García-Portugués <edgarcia@est-econ.uc3m.es>

Repository CRAN

Date/Publication 2026-11-12 21:40:02 UTC

Contents

rotasym-package . . . 2
ACG . . . 3
cosines-signs . . . 4
estimators . . . 7
sunspots . . . 8
tang-norm-decomp . . . 12
tangent-elliptical . . . 15
tangent-vMF . . . 18
test_rotasym . . . 20
unif . . . 25
vMF . . . 27

Index **30**

`rotasym-package` **rotasym** – *Tests for Rotational Symmetry on the Hypersphere*

Description

Implementation of the tests for rotational symmetry on the hypersphere proposed in García-Portugués, Paindaveine and Verdebout (2020). The package implements the proposed distributions on the hypersphere based on the tangent-normal decomposition. It also allows for the replication of the data application considered in the paper.

Author(s)

Eduardo García-Portugués, Davy Paindaveine, and Thomas Verdebout.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

Useful links:

- https://github.com/egarpor/rotasym
- Report bugs at https://github.com/egarpor/rotasym/issues

ACG *Angular central Gaussian distribution*

Description

Density and simulation of the Angular Central Gaussian (ACG) distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 1$. The density at $\boldsymbol{x} \in \mathcal{S}^{p-1}$, $p \geq 2$, is given by

$$c_{p,\boldsymbol{\Lambda}}^{\mathrm{ACG}}(\boldsymbol{x}'\boldsymbol{\Lambda}^{-1}\boldsymbol{x})^{-p/2} \quad \text{with} \quad c_{p,\boldsymbol{\Lambda}}^{\mathrm{ACG}} := 1/(\omega_p|\boldsymbol{\Lambda}|^{1/2})$$

where $\boldsymbol{\Lambda}$ is the shape matrix, a $p \times p$ symmetric and positive definite matrix, and ω_p is the surface area of $\mathcal{S}^{p-1}$.

Usage

```
d_ACG(x, Lambda, log = FALSE)

c_ACG(p, Lambda, log = FALSE)

r_ACG(n, Lambda)
```

Arguments

x	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size c(nx, p) or a vector of length p. Normalized internally if required (with a warning message).
Lambda	the shape matrix $\boldsymbol{\Lambda}$ of the ACG. A symmetric and positive definite matrix of size c(p, p).
log	compute the logarithm of the density (or the normalizing constant)? Defaults to FALSE.
p	dimension of the ambient space $\mathbb{R}^p$ that contains $\mathcal{S}^{p-1}$. A positive integer.
n	sample size, a positive integer.

Details

Due to the projection of the ACG, the shape matrix $\boldsymbol{\Lambda}$ is only identified up to a constant, that is, $\boldsymbol{\Lambda}$ and $c\boldsymbol{\Lambda}$ give the same ACG distribution. Usually, $\boldsymbol{\Lambda}$ is normalized to have trace equal to p.

c_ACG is vectorized on p. If $p = 1$, then the ACG is the uniform distribution in the set $\{-1, 1\}$.

Value

Depending on the function:

- d_ACG: a vector of length nx or 1 with the evaluated density at x.
- r_ACG: a matrix of size c(n, p) with the random sample.
- c_ACG: the normalizing constant.

References

Tyler, D. E. (1987). Statistical analysis for the angular central Gaussian distribution on the sphere. *Biometrika*, 74(3):579–589. doi:10.1093/biomet/74.3.579

See Also

tangent-elliptical, unif.

Examples

```
# Simulation and density evaluation for p = 2
Lambda <- diag(c(5, 1))
n <- 1e3
x <- r_ACG(n = n, Lambda = Lambda)
col <- viridisLite::viridis(n)
r <- runif(n, 0.95, 1.05) # Radius perturbation to improve visualization
dens <- d_ACG(x = x, Lambda = Lambda)
plot(r * x, pch = 16, col = col[rank(dens)])

# Simulation and density evaluation for p = 3
Lambda <- rbind(c(5, 1, 0.5),
                c(1, 2, 1),
                c(0.5, 1, 1))
x <- r_ACG(n = n, Lambda = Lambda)
dens <- d_ACG(x = x, Lambda = Lambda)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)
```

`cosines-signs` *Cosines and multivariate signs of a hyperspherical sample about a given location*

Description

Computation of the cosines and multivariate signs of the hyperspherical sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathcal{S}^{p-1}$ about a location $\boldsymbol{\theta} \in \mathcal{S}^{p-1}$, for $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$ with $p \geq 2$. The *cosines* are defined as

$$V_i := \boldsymbol{X}_i'\boldsymbol{\theta}, \quad i = 1, \ldots, n,$$

whereas the *multivariate signs* are the vectors $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n \in \mathcal{S}^{p-2}$ defined as

$$\boldsymbol{U}_i := \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{X}_i / \|\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{X}_i\|, \quad i = 1, \ldots, n.$$

The projection matrix $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ is a $p \times (p-1)$ semi-orthogonal matrix that satisfies

$$\boldsymbol{\Gamma}_{\boldsymbol{\theta}}'\boldsymbol{\Gamma}_{\boldsymbol{\theta}} = \boldsymbol{I}_{p-1} \quad \text{and} \quad \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}' = \boldsymbol{I}_p - \boldsymbol{\theta}\boldsymbol{\theta}'.$$

where $\boldsymbol{I}_p$ is the identity matrix of dimension p.

Usage

```
signs(X, theta, Gamma = NULL, check_X = FALSE)

cosines(X, theta, check_X = FALSE)

Gamma_theta(theta, eig = FALSE)

prod_Gamma_theta(X, theta, transpose = FALSE)
```

Arguments

X	hyperspherical data, a matrix of size c(n, p) with unit-norm rows. NAs are allowed. For prod_Gamma_theta, any matrix of size c(n, p), or c(n, p - 1) if transpose = TRUE. A vector is a single observation.
theta	a unit-norm vector of length p. Normalized internally if it does not have unit norm (with a warning message).
Gamma	output from Gamma_theta(theta = theta). If NULL (default), the product with $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ is computed without forming the $p \times (p-1)$ matrix.
check_X	check the unit norms on the rows of X? Defaults to FALSE for performance reasons.
eig	find $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ using an eigendecomposition of $\boldsymbol{I}_p - \boldsymbol{\theta}\boldsymbol{\theta}'$ (inefficient)? Defaults to FALSE.
transpose	multiply by $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}'$ instead of by $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$? Defaults to FALSE.

Details

Note that the projection matrix $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ is *not* unique. In particular, any completion of $\boldsymbol{\theta}$ to an orthonormal basis $\{\boldsymbol{\theta}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{p-1}\}$ gives a set of $p-1$ orthonormal p-vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{p-1}\}$ that conform the columns of $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$. If eig = FALSE, this approach is employed by rotating the canonical completion of $\boldsymbol{e}_1 = (1, 0, \ldots, 0)$, $\{\boldsymbol{e}_2, \ldots, \boldsymbol{e}_p\}$, by the rotation matrix that rotates $\boldsymbol{e}_1$ to $\boldsymbol{\theta}$:

$$\boldsymbol{H}_{\boldsymbol{\theta}} = (\boldsymbol{\theta} + \boldsymbol{e}_1)(\boldsymbol{\theta} + \boldsymbol{e}_1)'/(1 + \theta_1) - \boldsymbol{I}_p.$$

If eig = TRUE, then a much more expensive eigendecomposition of $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}' = \boldsymbol{I}_p - \boldsymbol{\theta}\boldsymbol{\theta}'$ is performed for determining $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{p-1}\}$.

prod_Gamma_theta computes the products $\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ and $\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}'$, with the $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ of eig = FALSE, without forming $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$. This saves the $O(p^2)$ memory of the $p \times (p-1)$ matrix, which is relevant for large dimensions p.

If signs and cosines are called with X without unit norms in the rows, then the results will be spurious. Setting check_X = TRUE prevents this from happening.

Value

Depending on the function:

- cosines: a vector of length n with the cosines of X.
- signs: a matrix of size c(n, p - 1) with the multivariate signs of X.

- `Gamma_theta`: a projection matrix $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ of size `c(p, p - 1)`.
- `prod_Gamma_theta`: a matrix of size `c(n, p - 1)` with $\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$, or of size `c(n, p)` with $\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}'$ if `transpose = TRUE`.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

`tang-norm-decomp`, `test_rotasym`.

Examples

```
# Gamma_theta
theta <- c(0, 1)
Gamma_theta(theta = theta)

# prod_Gamma_theta gives the same products without forming Gamma_theta
X <- r_unif_sphere(n = 5, p = 2)
prod_Gamma_theta(X = X, theta = theta) - X %*% Gamma_theta(theta = theta)

# Signs and cosines for p = 2
L <- rbind(c(1, 0.5),
           c(0.5, 1))
X <- r_ACG(n = 1e3, Lambda = L)
old_par <- par(mfrow = c(1, 2))
plot(signs(X = X, theta = theta), main = "Signs", xlab = expression(x[1]),
     ylab = expression(x[2]))
hist(cosines(X = X, theta = theta), prob = TRUE, main = "Cosines",
     xlab = expression(x * "'" * theta))
par(old_par)

# Signs and cosines for p = 3
L <- rbind(c(2, 0.25, 0.25),
           c(0.25, 0.5, 0.25),
           c(0.25, 0.25, 0.5))
X <- r_ACG(n = 1e3, Lambda = L)
old_par <- par(mfrow = c(1, 2))
theta <- c(0, 1, 0)
plot(signs(X = X, theta = theta), main = "Signs", xlab = expression(x[1]),
     ylab = expression(x[2]))
hist(cosines(X = X, theta = theta), prob = TRUE, main = "Cosines",
     xlab = expression(x * "'" * theta))
par(old_par)
```

`estimators` *Estimators for the axis of rotational symmetry* $\boldsymbol{\theta}$

Description

Estimation of the axis of rotational symmetry $\boldsymbol{\theta}$ of a rotationally symmetric unit-norm random vector $\boldsymbol{X}$ in $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 2$, from a hyperspherical sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathcal{S}^{p-1}$.

Usage

```
spherical_mean(data)

spherical_loc_PCA(data)
```

Arguments

`data`	hyperspherical data, a matrix of size `c(n, p)` with unit-norm rows. Normalized internally if any row does not have unit norm (with a `warning` message). `NA`s are ignored.

Details

The `spherical_mean` estimator computes the sample mean of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ and normalizes it by its norm, with an error if the norm is zero. It estimates consistently $\boldsymbol{\theta}$ for rotationally symmetric models based on angular functions g that are monotone increasing.

The estimator in `spherical_loc_PCA` is based on the fact that, under rotational symmetry, the expectation of $\boldsymbol{X}\boldsymbol{X}'$ is $a\boldsymbol{\theta}\boldsymbol{\theta}' + b(\boldsymbol{I}_p - \boldsymbol{\theta}\boldsymbol{\theta}')$ for certain constants $a, b \geq 0$. Therefore, $\boldsymbol{\theta}$ is the eigenvector with unique multiplicity of the expectation of $\boldsymbol{X}\boldsymbol{X}'$. Its use is recommended if the rotationally symmetric data is not unimodal.

Value

A vector of length `p` with an estimate for $\boldsymbol{\theta}$.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

`test_rotasym`, `cosines-signs`.

Examples

```
# Sample from a vMF
n <- 200
p <- 10
theta <- c(1, rep(0, p - 1))
set.seed(123456789)
data <- r_vMF(n = n, mu = theta, kappa = 3)
theta_mean <- spherical_mean(data)
theta_PCA <- spherical_loc_PCA(data)
sqrt(sum((theta - theta_mean)^2)) # More efficient
sqrt(sum((theta - theta_PCA)^2))

# Sample from a mixture of antipodal vMF's
n <- 200
p <- 10
theta <- c(1, rep(0, p - 1))
set.seed(123456789)
data <- rbind(r_vMF(n = n, mu = theta, kappa = 3),
              r_vMF(n = n, mu = -theta, kappa = 3))
theta_mean <- spherical_mean(data)
theta_PCA <- spherical_loc_PCA(data)
sqrt(sum((theta - theta_mean)^2))
sqrt(sum((theta - theta_PCA)^2)) # Better suited in this case
```

---

sunspots *Recorded sunspots births/deaths during 1872–2018*

Description

Processed version of the Debrecen Photoheliographic Data (DPD) sunspot catalog and the revised version of the Greenwich Photoheliographic Results (GPR) sunspot catalog. The two sources contain the records of sunspots appeared during 1872–2018 (GPR for 1872–1976; DPD for 1974–2018).

Sunspots appear in groups and have a variable lifetime. This dataset has been processed to account for the *births* (first observations) and *deaths* (last observations) of *groups* of sunspots.

Usage

```
sunspots_births

sunspots_deaths
```

Format

A data frame with 51303 rows and 7 variables:

NOAA NOAA identifier of the group of sunspots, a factor.

date UTC date, as `POSIXct`, of the first (`sunspots_births`) or last (`sunspots_deaths`) observation of a group of sunspots.

cycle solar cycle in which the group of sunspots was observed.

total_area total whole spot area of the group, measured in millionths of the solar hemisphere.

dist_sun_disc distance from the center of Sun's disc, measured in units of the solar radius.

theta mean longitude angle $\theta \in [0, 2\pi)$ of the group position.

phi mean latitude angle $\phi \in [-\pi/2, \pi/2)$ of the group position.

Details

The mean position of the group of sunspots is obtained by a weighted average of the positions of the single sunspots by the whole spot area of the single spots. The areas are corrected to account for foreshortening.

The (θ, ϕ) angles are such their associated Cartesian coordinates are:

$$(\cos(\phi)\cos(\theta), \cos(\phi)\sin(\theta), \sin(\phi)),$$

with $(0, 0, 1)$ denoting the north pole.

The DPD data has different states of completeness and quality control. The longest span of "final complete data" (no missing observation days and the data has undergone a systematic quality control) is from 2005 to 2015.

The data has been preprocessed using the following pipeline:

1. Retrieve data from the GPR and DPD sunspot catalogs.
2. Omit observations with NAs in the sunspot positions.
3. Filter for sunspot groups.
4. Relabel the NOAA identifier for the sunspot group for records before 1974, prefixing the "GPR" string. Otherwise, very different groups of sunspots from the two catalogs may share the same identifier.
5. Keep only the first (births) or last (deaths) row of each NOAA instance, that is, the first-ever or last-ever observation of each sunspot group.

The script performing the preprocessing is available at `sunspots.R`

Source

`http://fenyi.solarobs.epss.hun-ren.hu`

References

Baranyi, T., Győri, L., and Ludmány, A. (2016). On-line tools for solar data compiled at the Debrecen observatory and their extensions with the Greenwich sunspot data. *Solar Physics*, 291(9–10):3081–3102. doi:10.1007/s1120701609301

Győri, L., Ludmány, A., and Baranyi, T. (2017). Comparative analysis of Debrecen sunspot catalogues. *Monthly Notices of the Royal Astronomical Society*, 465(2):1259–1273. doi:10.1093/mnras/stw2667

See Also

`test_rotasym`, `cosines-signs`.

Examples

```
# Load births data of the 23rd cycle
data("sunspots_births")
sp_bir_23 <- subset(sunspots_births, cycle == 23)

# Transform to Cartesian coordinates
sp_bir_23$X_bir <-
  cbind(cos(sp_bir_23$phi) * cos(sp_bir_23$theta),
        cos(sp_bir_23$phi) * sin(sp_bir_23$theta),
        sin(sp_bir_23$phi))

# Plot births
n_cols <- 100
cuts <- cut(x = sp_bir_23$date, include.lowest = TRUE,
            breaks = quantile(sp_bir_23$date,
                              probs = seq(0, 1, l = n_cols + 1)))
scatterplot3d::scatterplot3d(sp_bir_23$X_bir, xlim = c(-1, 1),
                             ylim = c(-1, 1), zlim = c(-1, 1),
                             color = viridisLite::viridis(n_cols)[cuts],
                             pch = 16, cex.symbols = 0.5, xlab = "",
                             ylab = "", zlab = "", angle = 30)
# Spörer's law: sunspots at the beginning of the solar cycle (dark blue
# color) tend to appear at higher latitudes, gradually decreasing to the
# equator as the solar cycle advances (yellow color).

# Estimation of the density of the cosines
V <- cosines(X = sp_bir_23$X_bir, theta = c(0, 0, 1))
h <- bw.SJ(x = V, method = "dpi")
plot(kde <- density(x = V, bw = h, n = 2^13, from = -1, to = 1), col = 1,
     xlim = c(-1, 1), ylim = c(0, 3), axes = FALSE, main = "",
     xlab = "Cosines (latitude angles)", lwd = 2)
at <- seq(-1, 1, by = 0.25)
axis(2); axis(1, at = at)
axis(1, at = at, line = 1, tick = FALSE,
     labels = paste0("(", 90 - round(acos(at) / pi * 180, 1), "º)"))
rug(V)
legend("topright", legend = c("Full cycle", "Initial 25% cycle",
                              "Final 25% cycle"),
       lwd = 2, col = c(1, viridisLite::viridis(12)[c(3, 8)]))

# Density for the observations within the initial 25% of the cycle
part1 <- sp_bir_23$date < quantile(sp_bir_23$date, 0.25)
V1 <- cosines(X = sp_bir_23$X[part1, ], theta = c(0, 0, 1))
h1 <- bw.SJ(x = V1, method = "dpi")
lines(kde1 <- density(x = V1, bw = h1, n = 2^13, from = -1, to = 1),
      col = viridisLite::viridis(12)[3], lwd = 2)

# Density for the observations within the final 25% of the cycle
part2 <- sp_bir_23$date > quantile(sp_bir_23$date, 0.75)
V2 <- cosines(X = sp_bir_23$X[part2, ], theta = c(0, 0, 1))
h2 <- bw.SJ(x = V2, method = "dpi")
lines(kde2 <- density(x = V2, bw = h2, n = 2^13, from = -1, to = 1),
```

```
      col = viridisLite::viridis(12)[8], lwd = 2)

# Computation the level set of a kernel density estimator that contains
# at least 1 - alpha of the probability (kde stands for an object
# containing the output of density(x = data)).
kde_level_set <- function(kde, data, alpha) {

  # Estimate c from alpha
  c <- quantile(approx(x = kde$x, y = kde$y, xout = data)$y, probs = alpha)

  # Begin and end index for the potentially many intervals in the level sets
  kde_larger_c <- kde$y >= c
  run_length_kde <- rle(kde_larger_c)
  begin <- which(diff(kde_larger_c) > 0) + 1
  end <- begin + run_length_kde$lengths[run_length_kde$values] - 1

  # Return the [a_i, b_i], i = 1, ..., K in the K rows
  return(cbind(kde$x[begin], kde$x[end]))

}

# Level set containing the 90% of the probability, in latitude angles
90 - acos(kde_level_set(kde = kde, data = V, alpha = 0.10)) / pi * 180

# Modes (in cosines and latitude angles)
modes <- c(kde$x[kde$x < 0][which.max(kde$y[kde$x < 0])],
           kde$x[kde$x > 0][which.max(kde$y[kde$x > 0])])
90 - acos(modes) / pi * 180

# Load deaths data of the 23rd cycle
data("sunspots_deaths")
sp_dea_23 <- subset(sunspots_deaths, cycle == 23)

# Transform to Cartesian coordinates
sp_dea_23$X_dea <-
  cbind(cos(sp_dea_23$phi) * cos(sp_dea_23$theta),
        cos(sp_dea_23$phi) * sin(sp_dea_23$theta),
        sin(sp_dea_23$phi))

# Match births and deaths, and exclude single-day observations
sp_23 <- dplyr::full_join(sp_bir_23, sp_dea_23, by = "NOAA",
                          suffix = c("_bir", "_dea"))
sp_23 <- sp_23[sp_23$date_bir != sp_23$date_dea, ]

# Plot births and deaths associated to the 23rd cycle, and color according
# to movement direction of sunspots.
sc <- scatterplot3d::scatterplot3d(sp_23$X_bir, xlim = c(-1, 1),
                                   ylim = c(-1, 1), zlim = c(-1, 1),
                                   color = 1, cex.symbols = 0.1,
                                   pch = 16, xlab = "", ylab = "", zlab = "",
                                   angle = 30)
sc$points3d(sp_23$X_dea, col = 1, cex = 0.1)
for (i in seq_len(nrow(sp_23))) {
```

```
  sc$points3d(rbind(sp_23$X_bir[i, ], sp_23$X_dea[i, ]),
              col = ifelse((sp_23$theta_bir[i] - sp_23$theta_dea[i]) %%
                           (2 * pi) < pi, 2, 3), type = "l")
}
```

---

tang-norm-decomp *Distributions based on the tangent-normal decomposition*

Description

Density and simulation of a distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 2$, obtained by the tangent-normal decomposition. The *tangent-normal decomposition* of the random vector $\boldsymbol{X} \in \mathcal{S}^{p-1}$ is

$$V\boldsymbol{\theta} + \sqrt{1 - V^2}\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{U}$$

where $V := \boldsymbol{X}'\boldsymbol{\theta}$ is a random variable in $[-1, 1]$ (the *cosines* of $\boldsymbol{X}$) and $\boldsymbol{U} := \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{X}/\|\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{X}\|$ is a random vector in $\mathcal{S}^{p-2}$ (the *multivariate signs* of $\boldsymbol{X}$) and $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$ is the $p \times (p-1)$ matrix computed by `Gamma_theta`.

The tangent-normal decomposition can be employed for constructing distributions for $\boldsymbol{X}$ that arise for certain choices of V and $\boldsymbol{U}$. If V and $\boldsymbol{U}$ are *independent*, then simulation from $\boldsymbol{X}$ is straightforward using the tangent-normal decomposition. Also, the density of $\boldsymbol{X}$ at $\boldsymbol{x} \in \mathcal{S}^{p-1}$, $f_X(\boldsymbol{x})$, is readily computed as

$$f_X(\boldsymbol{x}) = \omega_{p-1} c_g g(t)(1 - t^2)^{(p-3)/2} f_U(\boldsymbol{u})$$

where $t := \boldsymbol{x}'\boldsymbol{\theta}$, $\boldsymbol{u} := \boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{x}/\|\boldsymbol{\Gamma}_{\boldsymbol{\theta}}\boldsymbol{x}\|$, f_U is the density of $\boldsymbol{U}$, and $f_V(v) := \omega_{p-1} c_g g(v)(1 - v^2)^{(p-3)/2}$ is the density of V for an angular function g with normalizing constant c_g. ω_{p-1} is the surface area of $\mathcal{S}^{p-2}$.

Usage

```
d_tang_norm(x, theta, g_scaled, d_V, d_U, log = FALSE)

r_tang_norm(n, theta, r_U, r_V)
```

Arguments

`x`	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size `c(nx, p)` or a vector of length `p`. Normalized internally if required (with a `warning` message).
`theta`	a unit-norm vector of length `p` giving the axis of rotational symmetry. Normalized internally if it does not have unit norm (with a `warning` message).
`g_scaled`	the *scaled* angular density $c_g g$. In the form `g_scaled <- function(t, log = TRUE) {...}`. See examples.
`d_V`	the density f_V. In the form `d_V <- function(v, log = TRUE) {...}`. See examples.
`d_U`	the density $f_{\boldsymbol{U}}$. In the form `d_U <- function(u, log = TRUE) {...}`. See examples.

log	compute the logarithm of the density (or the normalizing constant)? Defaults to FALSE.
n	sample size, a positive integer.
r_U	a function for simulating $\boldsymbol{U}$. Its first argument must be the sample size. See examples.
r_V	a function for simulating V. Its first argument must be the sample size. See examples.

Details

Either g_scaled or d_V can be supplied to d_tang_norm (the rest of the arguments are compulsory). One possible choice for g_scaled is g_vMF with scaled = TRUE. Another possible choice is the angular function $g(t) = 1-t^2$, normalized by its normalizing constant $c_g = (\Gamma(p/2)p)/(2\pi^{p/2}(p-1))$ (see examples). This angular function makes V^2 to be distributed as a $\mathrm{Beta}(1/2, (p+1)/2)$.

The normalizing constants and densities are computed through log-scales for numerical accuracy.

Value

Depending on the function:

- d_tang_norm: a vector of length nx or 1 with the evaluated density at x.
- r_tang_norm: a matrix of size c(n, p) with the random sample.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

cosines-signs, tangent-elliptical, tangent-vMF.

Examples

```
## Simulation and density evaluation for p = 2

# Parameters
n <- 1e3
p <- 2
theta <- c(rep(0, p - 1), 1)
mu <- c(rep(0, p - 2), 1)
kappa_V <- 2
kappa_U <- 0.1

# The vMF scaled angular function
g_scaled <- function(t, log) {
  g_vMF(t, p = p, kappa = kappa_V, scaled = TRUE, log = log)
}
```

```
# Cosine density for the vMF distribution
d_V <- function(v, log) {
  log_dens <- w_p(p = p - 1, log = TRUE) + g_scaled(t = v, log = TRUE) +
    0.5 * (p - 3) * log(1 - v^2)
  switch(log + 1, exp(log_dens), log_dens)
}

# Multivariate signs density based on a vMF
d_U <- function(x, log) d_vMF(x = x, mu = mu, kappa = kappa_U, log = log)

# Simulation functions
r_V <- function(n) r_g_vMF(n = n, p = p, kappa = kappa_V)
r_U <- function(n) r_vMF(n = n, mu = mu, kappa = kappa_U)

# Sample and color according to density
x <- r_tang_norm(n = n, theta = theta, r_V = r_V, r_U = r_U)
r <- runif(n, 0.95, 1.05) # Radius perturbation to improve visualization
col <- viridisLite::viridis(n)
dens <- d_tang_norm(x = x, theta = theta, g_scaled = g_scaled, d_U = d_U)
# dens <- d_tang_norm(x = x, theta = theta, d_V = d_V, d_U = d_U) # The same
plot(r * x, pch = 16, col = col[rank(dens)])

## Simulation and density evaluation for p = 3

# Parameters
p <- 3
n <- 5e3
theta <- c(rep(0, p - 1), 1)
mu <- c(rep(0, p - 2), 1)
kappa_V <- 2
kappa_U <- 2

# Sample and color according to density
x <- r_tang_norm(n = n, theta = theta, r_V = r_V, r_U = r_U)
col <- viridisLite::viridis(n)
dens <- d_tang_norm(x = x, theta = theta, g_scaled = g_scaled, d_U = d_U)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)

## A non-vMF angular function: g(t) = 1 - t^2. It is associated to the
## Beta(1/2, (p + 1)/2) distribution.

# Scaled angular function
g_scaled <- function(t, log) {
  log_c_g <- lgamma(0.5 * p) + log(0.5 * p / (p - 1)) - 0.5 * p * log(pi)
  log_g <- log_c_g + log(1 - t^2)
  switch(log + 1, exp(log_g), log_g)
}

# Cosine density
d_V <- function(v, log) {
```

```
  log_dens <- w_p(p = p - 1, log = TRUE) + g_scaled(t = v, log = TRUE) +
    (0.5 * (p - 3)) * log(1 - v^2)
  switch(log + 1, exp(log_dens), log_dens)
}

# Simulation
r_V <- function(n) {
  sample(x = c(-1, 1), size = n, replace = TRUE) *
    sqrt(rbeta(n = n, shape1 = 0.5, shape2 = 0.5 * (p + 1)))
}

# Sample and color according to density
r_U <- function(n) r_unif_sphere(n = n, p = p - 1)
x <- r_tang_norm(n = n, theta = theta, r_V = r_V, r_U = r_U)
col <- viridisLite::viridis(n)
dens <- d_tang_norm(x = x, theta = theta, d_V = d_V, d_U = d_unif_sphere)
# dens <- d_tang_norm(x = x, theta = theta, g_scaled = g_scaled,
#                     d_U = d_unif_sphere) # The same
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)
```

`tangent-elliptical` *Tangent elliptical distribution*

Description

Density and simulation of the Tangent Elliptical (TE) distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 2$. The distribution arises by considering the tangent-normal decomposition with multivariate signs distributed as an Angular Central Gaussian distribution.

Usage

```
d_TE(x, theta, g_scaled, d_V, Lambda, log = FALSE)

r_TE(n, theta, r_V, Lambda)
```

Arguments

`x`	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size `c(nx, p)` or a vector of length `p`. Normalized internally if required (with a `warning` message).
`theta`	a unit-norm vector of length `p` giving the axis of rotational symmetry. Normalized internally if it does not have unit norm (with a `warning` message).
`g_scaled`	the *scaled* angular density $c_g g$. In the form `g_scaled <- function(t, log = TRUE) {...}`. See examples.
`d_V`	the density f_V. In the form `d_V <- function(v, log = TRUE) {...}`. See examples.

Lambda	the shape matrix $\mathbf{\Lambda}$ of the ACG used in the multivariate signs. A symmetric and positive definite matrix of size c(p - 1, p - 1).
log	compute the logarithm of the density (or the normalizing constant)? Defaults to FALSE.
n	sample size, a positive integer.
r_V	a function for simulating V. Its first argument must be the sample size. See examples.

Details

The functions are wrappers for d_tang_norm and r_tang_norm with d_U = d_ACG and r_U = r_ACG.

Value

Depending on the function:

- d_TE: a vector of length nx or 1 with the evaluated density at x.
- r_TE: a matrix of size c(n, p) with the random sample.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

tang-norm-decomp, ACG, test_rotasym.

Examples

```
## Simulation and density evaluation for p = 2

# Parameters
p <- 2
n <- 1e3
theta <- c(rep(0, p - 1), 1)
Lambda <- matrix(0.5, nrow = p - 1, ncol = p - 1)
diag(Lambda) <- 1
kappa_V <- 2

# Required functions
r_V <- function(n) r_g_vMF(n = n, p = p, kappa = kappa_V)
g_scaled <- function(t, log) {
  g_vMF(t, p = p, kappa = kappa_V, scaled = TRUE, log = log)
}

# Sample and color according to density
x <- r_TE(n = n, theta = theta, r_V = r_V, Lambda = Lambda)
col <- viridisLite::viridis(n)
```

```
r <- runif(n, 0.95, 1.05) # Radius perturbation to improve visualization
dens <- d_TE(x = x, theta = theta, g_scaled = g_scaled, Lambda = Lambda)
plot(r * x, pch = 16, col = col[rank(dens)])

## Simulation and density evaluation for p = 3

# Parameters
p <- 3
n <- 5e3
theta <- c(rep(0, p - 1), 1)
Lambda <- matrix(0.5, nrow = p - 1, ncol = p - 1)
diag(Lambda) <- 1
kappa_V <- 2

# Sample and color according to density
x <- r_TE(n = n, theta = theta, r_V = r_V, Lambda = Lambda)
col <- viridisLite::viridis(n)
dens <- d_TE(x = x, theta = theta, g_scaled = g_scaled, Lambda = Lambda)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)

## A non-vMF angular function: g(t) = 1 - t^2. It is associated to the
## Beta(1/2, (p + 1)/2) distribution.

# Scaled angular function
g_scaled <- function(t, log) {
  log_c_g <- lgamma(0.5 * p) + log(0.5 * p / (p - 1)) - 0.5 * p * log(pi)
  log_g <- log_c_g + log(1 - t^2)
  switch(log + 1, exp(log_g), log_g)
}

# Simulation
r_V <- function(n) {
  sample(x = c(-1, 1), size = n, replace = TRUE) *
    sqrt(rbeta(n = n, shape1 = 0.5, shape2 = 0.5 * (p + 1)))
}

# Sample and color according to density
kappa_V <- 1
Lambda <- matrix(0.75, nrow = p - 1, ncol = p - 1)
diag(Lambda) <- 1
x <- r_TE(n = n, theta = theta, r_V = r_V, Lambda = Lambda)
col <- viridisLite::viridis(n)
dens <- d_TE(x = x, theta = theta, g_scaled = g_scaled, Lambda = Lambda)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)
```

tangent-vMF *Tangent von Mises–Fisher distribution*

Description

Density and simulation of the Tangent von Mises–Fisher (TM) distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 2$. The distribution arises by considering the tangent-normal decomposition with multivariate signs distributed as a von Mises–Fisher distribution.

Usage

```
d_TM(x, theta, g_scaled, d_V, mu, kappa, log = FALSE)

r_TM(n, theta, r_V, mu, kappa)
```

Arguments

x	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size c(nx, p) or a vector of length p. Normalized internally if required (with a warning message).
theta	a unit-norm vector of length p giving the axis of rotational symmetry. Normalized internally if it does not have unit norm (with a warning message).
g_scaled	the *scaled* angular density $c_g g$. In the form g_scaled <- function(t, log = TRUE) {...}. See examples.
d_V	the density f_V. In the form d_V <- function(v, log = TRUE) {...}. See examples.
mu	the directional mean $\boldsymbol{\mu}$ of the vMF used in the multivariate signs. A unit-norm vector of length p - 1.
kappa	concentration parameter κ of the vMF used in the multivariate signs. A nonnegative scalar.
log	compute the logarithm of the density (or the normalizing constant)? Defaults to FALSE.
n	sample size, a positive integer.
r_V	a function for simulating V. Its first argument must be the sample size. See examples.

Details

The functions are wrappers for d_tang_norm and r_tang_norm with d_U = d_vMF and r_U = r_vMF.

Value

Depending on the function:

- d_TM: a vector of length nx or 1 with the evaluated density at x.
- r_TM: a matrix of size c(n, p) with the random sample.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

tang-norm-decomp, vMF, test_rotasym.

Examples

```
## Simulation and density evaluation for p = 2

# Parameters
p <- 2
n <- 1e3
theta <- c(rep(0, p - 1), 1)
mu <- c(rep(0, p - 2), 1)
kappa <- 1
kappa_V <- 2

# Required functions
r_V <- function(n) r_g_vMF(n = n, p = p, kappa = kappa_V)
g_scaled <- function(t, log) {
  g_vMF(t, p = p, kappa = kappa_V, scaled = TRUE, log = log)
}

# Sample and color according to density
x <- r_TM(n = n, theta = theta, r_V = r_V, mu = mu, kappa = kappa)
col <- viridisLite::viridis(n)
r <- runif(n, 0.95, 1.05) # Radius perturbation to improve visualization
dens <- d_TM(x = x, theta = theta, g_scaled = g_scaled, mu = mu,
             kappa = kappa)
plot(r * x, pch = 16, col = col[rank(dens)])

## Simulation and density evaluation for p = 3

# Parameters
p <- 3
n <- 5e3
theta <- c(rep(0, p - 1), 1)
mu <- c(rep(0, p - 2), 1)
kappa <- 1
kappa_V <- 2

# Sample and color according to density
x <- r_TM(n = n, theta = theta, r_V = r_V, mu = mu, kappa = kappa)
col <- viridisLite::viridis(n)
dens <- d_TM(x = x, theta = theta, g_scaled = g_scaled, mu = mu,
             kappa = kappa)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
```

```
                              pch = 16, xlab = "", ylab = "", zlab = "",
                              angle = 20)

## A non-vMF angular function: g(t) = 1 - t^2. It is associated to the
## Beta(1/2, (p + 1)/2) distribution.

# Scaled angular function
g_scaled <- function(t, log) {
  log_c_g <- lgamma(0.5 * p) + log(0.5 * p / (p - 1)) - 0.5 * p * log(pi)
  log_g <- log_c_g + log(1 - t^2)
  switch(log + 1, exp(log_g), log_g)
}

# Simulation
r_V <- function(n) {
  sample(x = c(-1, 1), size = n, replace = TRUE) *
    sqrt(rbeta(n = n, shape1 = 0.5, shape2 = 0.5 * (p + 1)))
}

# Sample and color according to density
kappa <- 0.5
x <- r_TM(n = n, theta = theta, r_V = r_V, mu = mu, kappa = kappa)
col <- viridisLite::viridis(n)
dens <- d_TM(x = x, theta = theta, g_scaled = g_scaled,
             mu = mu, kappa = kappa)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                              zlim = c(-1, 1), color = col[rank(dens)],
                              pch = 16, xlab = "", ylab = "", zlab = "",
                              angle = 20)
```

`test_rotasym` *Tests of rotational symmetry for hyperspherical data*

Description

Tests for assessing the rotational symmetry of a unit-norm random vector $\boldsymbol{X}$ in $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 2$, about a location $\boldsymbol{\theta} \in \mathcal{S}^{p-1}$, from a hyperspherical sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathcal{S}^{p-1}$.

The vector $\boldsymbol{X}$ is said to be rotationally symmetric about $\boldsymbol{\theta}$ if the distributions of $\boldsymbol{O}\boldsymbol{X}$ and $\boldsymbol{X}$ coincide, where $\boldsymbol{O}$ is any $p \times p$ rotation matrix that fixes $\boldsymbol{\theta}$, i.e., $\boldsymbol{O}\boldsymbol{\theta} = \boldsymbol{\theta}$.

Usage

```
test_rotasym(
  data,
  theta = spherical_mean,
  type = c("sc", "loc", "loc_vMF", "hyb", "hyb_vMF")[5],
  Fisher = FALSE,
  U = NULL,
  V = NULL
)
```

Arguments

`data`	hyperspherical data, a matrix of size `c(n, p)` with unit-norm rows. Normalized internally if any row does not have unit norm (with a `warning` message). `NA`s are ignored.
`theta`	either a unit-norm vector of length `p` giving the axis of rotational symmetry (for the specified-$\boldsymbol{\theta}$ case) or a function that implements an estimator $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ (for the unspecified-$\boldsymbol{\theta}$ case). The default calls the `spherical_mean` function. A vector is normalized internally if it does not have unit norm (with a `warning` message). See examples.
`type`	a character string (case insensitive) indicating the type of test to conduct: • `"sc"`: "scatter" test based on the statistic $Q^{\text{sc}}_{\boldsymbol{\theta}}$. Evaluates if the covariance matrix of the multivariate signs is isotropic. • `"loc"`: "location" test based on the statistic $Q^{\text{loc}}_{\boldsymbol{\theta}}$. Evaluates if the expectation of the multivariate signs is zero. • `"loc_vMF"`: adapted "location" test, based on the statistic $Q^{\text{loc}}_{\text{vMF}}$. • `"hyb"`: "hybrid" test based on the statistics $Q^{\text{sc}}_{\boldsymbol{\theta}}$ and $Q^{\text{loc}}_{\boldsymbol{\theta}}$. • `"hyb_vMF"` (default): adapted "hybrid" test based on the statistics $Q^{\text{sc}}_{\boldsymbol{\theta}}$ and $Q^{\text{loc}}_{\text{vMF}}$. See the details below for further explanations of the tests.
`Fisher`	use Fisher's method to aggregate the scatter and location tests in the hybrid test? Otherwise, the hybrid statistic is the sum of the scatter and location statistics. See details below. Defaults to `FALSE`.
`U`	*multivariate signs* of `data`, a matrix of size `c(n, p - 1)`. Computed if `NULL` (the default).
`V`	*cosines* of `data`, a vector of length `n`. Computed if `NULL` (the default).

Details

Descriptions of the tests:

- The "scatter" test is locally and asymptotically optimal against tangent elliptical alternatives to rotational symmetry. However, it is not consistent against tangent von Mises–Fisher (vMF) alternatives. The asymptotic null distribution of $Q^{\text{sc}}_{\boldsymbol{\theta}}$ is unaffected if $\boldsymbol{\theta}$ is estimated, that is, the asymptotic null distributions of $Q^{\text{sc}}_{\boldsymbol{\theta}}$ and $Q^{\text{sc}}_{\hat{\boldsymbol{\theta}}}$ are the same.
- The "location" test is locally and asymptotically most powerful against vMF alternatives to rotational symmetry. However, it is not consistent against tangent elliptical alternatives. The asymptotic null distribution of $Q^{\text{loc}}_{\boldsymbol{\theta}}$ for known $\boldsymbol{\theta}$ (the one implemented in `test_rotasym`) *does change* if $\boldsymbol{\theta}$ is estimated by $\hat{\boldsymbol{\theta}}$. Therefore, if the test is performed with an estimated $\boldsymbol{\theta}$ (if `theta` is a function) $Q^{\text{loc}}_{\hat{\boldsymbol{\theta}}}$ will not be properly calibrated. `test_rotasym` will give a warning in such case.
- The "vMF location" test is a modification of the "location" test designed to make its null asymptotic distribution invariant from the estimation of $\boldsymbol{\theta}$ (as the "scatter" test is). The test is optimal against tangent vMF alternatives with a *specific*, vMF-based, angular function `g_vMF`. Despite not being optimal against all tangent vMF alternatives, it is consistent for all of them. As the location test, it is not consistent against tangent elliptical alternatives.

- The "hybrid" test combines (see below how) the "scatter" and "location" tests. The test is neither optimal against tangent elliptical nor tangent vMF alternatives, but it is consistent against both. Since it is based on the "location" test, if computed with an estimator $\hat{\boldsymbol{\theta}}$, the test statistic will not be properly calibrated. `test_rotasym` will give a warning in such case.
- The "vMF hybrid" test is the analogous of the "hybrid" test but replaces the "location" test by the "vMF location" test.

The combination of the scatter and location tests in the hybrid tests is done in two different ways:

- If `Fisher = FALSE`, then the scatter and location tests statistics give the hybrid test statistic

$$Q^{\text{hyb}}_{\boldsymbol{\theta}} := Q^{\text{sc}}_{\boldsymbol{\theta}} + Q^{\text{loc}}_{\boldsymbol{\theta}}.$$

- If `Fisher = TRUE`, then Fisher's method for aggregating independent tests (the two test statistics are independent under rotational symmetry) is considered, resulting the hybrid test statistic:

$$Q^{\text{hyb}}_{\boldsymbol{\theta}} := -2(\log(p_{\text{sc}}) + \log(p_{\text{loc}}))$$

where p_{sc} and p_{loc} are the p-values of the scatter and location tests, respectively.

The hybrid test statistic $Q^{\text{hyb}}_{\text{vMF}}$ follows analogously to $Q^{\text{hyb}}_{\boldsymbol{\theta}}$ by replacing $Q^{\text{loc}}_{\boldsymbol{\theta}}$ with $Q^{\text{loc}}_{\text{vMF}}$.

Finally, recall that the tests are designed to test *implications* of rotational symmetry. Therefore, the tests are not consistent against *all* types of alternatives to rotational symmetry.

Value

An object of the `htest` class with the following elements:

- `statistic`: test statistic.
- `parameter`: degrees of freedom of the chi-square distribution appearing in all the null asymptotic distributions.
- `p.value`: p-value of the test.
- `method`: information on the type of test performed.
- `data.name`: name of the value of `data`.
- `U`: multivariate signs of `data`.
- `V`: cosines of `data`.

References

García-Portugués, E., Paindaveine, D., and Verdebout, T. (2020). On optimal tests for rotational symmetry against new classes of hyperspherical distributions. *Journal of the American Statistical Association*, 115(532):1873–1887. doi:10.1080/01621459.2019.1665527

See Also

`tangent-elliptical`, `tangent-vMF`, `estimators`.

Examples

```
## Rotational symmetry holds

# Sample data from a vMF (rotationally symmetric distribution about mu)
n <- 200
p <- 10
theta <- c(1, rep(0, p - 1))
set.seed(123456789)
data_0 <- r_vMF(n = n, mu = theta, kappa = 1)

# theta known
test_rotasym(data = data_0, theta = theta, type = "sc")
test_rotasym(data = data_0, theta = theta, type = "loc")
test_rotasym(data = data_0, theta = theta, type = "loc_vMF")
test_rotasym(data = data_0, theta = theta, type = "hyb")
test_rotasym(data = data_0, theta = theta, type = "hyb", Fisher = TRUE)
test_rotasym(data = data_0, theta = theta, type = "hyb_vMF")
test_rotasym(data = data_0, theta = theta, type = "hyb_vMF", Fisher = TRUE)

# theta unknown (employs the spherical mean as estimator)
test_rotasym(data = data_0, type = "sc")
test_rotasym(data = data_0, type = "loc") # Warning
test_rotasym(data = data_0, type = "loc_vMF")
test_rotasym(data = data_0, type = "hyb") # Warning
test_rotasym(data = data_0, type = "hyb", Fisher = TRUE) # Warning
test_rotasym(data = data_0, type = "hyb_vMF")
test_rotasym(data = data_0, type = "hyb_vMF", Fisher = TRUE)

## Rotational symmetry does not hold

# Sample non-rotationally symmetric data from a tangent-vMF distribution.
# The scatter test is blind to these deviations, while the location tests
# are optimal.
n <- 200
p <- 10
theta <- c(1, rep(0, p - 1))
mu <- c(rep(0, p - 2), 1)
kappa <- 2
set.seed(123456789)
r_V <- function(n) {
  r_g_vMF(n = n, p = p, kappa = 1)
}
data_1 <- r_TM(n = n, r_V = r_V, theta = theta, mu = mu, kappa = kappa)

# theta known
test_rotasym(data = data_1, theta = theta, type = "sc")
test_rotasym(data = data_1, theta = theta, type = "loc")
test_rotasym(data = data_1, theta = theta, type = "loc_vMF")
test_rotasym(data = data_1, theta = theta, type = "hyb")
test_rotasym(data = data_1, theta = theta, type = "hyb", Fisher = TRUE)
test_rotasym(data = data_1, theta = theta, type = "hyb_vMF")
test_rotasym(data = data_1, theta = theta, type = "hyb_vMF", Fisher = TRUE)
```

```
# theta unknown (employs the spherical mean as estimator)
test_rotasym(data = data_1, type = "sc")
test_rotasym(data = data_1, type = "loc") # Warning
test_rotasym(data = data_1, type = "loc_vMF")
test_rotasym(data = data_1, type = "hyb") # Warning
test_rotasym(data = data_1, type = "hyb", Fisher = TRUE) # Warning
test_rotasym(data = data_1, type = "hyb_vMF")
test_rotasym(data = data_1, type = "hyb_vMF", Fisher = TRUE)

# Sample non-rotationally symmetric data from a tangent-elliptical
# distribution. The location tests are blind to these deviations, while the
# scatter test is optimal.
n <- 200
p <- 10
theta <- c(1, rep(0, p - 1))
Lambda <- matrix(0.5, nrow = p - 1, ncol = p - 1)
diag(Lambda) <- 1
set.seed(123456789)
r_V <- function(n) {
  r_g_vMF(n = n, p = p, kappa = 1)
}
data_2 <- r_TE(n = n, r_V = r_V, theta = theta, Lambda = Lambda)

# theta known
test_rotasym(data = data_2, theta = theta, type = "sc")
test_rotasym(data = data_2, theta = theta, type = "loc")
test_rotasym(data = data_2, theta = theta, type = "loc_vMF")
test_rotasym(data = data_2, theta = theta, type = "hyb")
test_rotasym(data = data_2, theta = theta, type = "hyb", Fisher = TRUE)
test_rotasym(data = data_2, theta = theta, type = "hyb_vMF")
test_rotasym(data = data_2, theta = theta, type = "hyb_vMF", Fisher = TRUE)

# theta unknown (employs the spherical mean as estimator)
test_rotasym(data = data_2, type = "sc")
test_rotasym(data = data_2, type = "loc") # Warning
test_rotasym(data = data_2, type = "loc_vMF")
test_rotasym(data = data_2, type = "hyb") # Warning
test_rotasym(data = data_2, type = "hyb", Fisher = TRUE) # Warning
test_rotasym(data = data_2, type = "hyb_vMF")
test_rotasym(data = data_2, type = "hyb_vMF", Fisher = TRUE)

## Sunspots births data

# Load data
data("sunspots_births")
sunspots_births$X <-
  cbind(cos(sunspots_births$phi) * cos(sunspots_births$theta),
        cos(sunspots_births$phi) * sin(sunspots_births$theta),
        sin(sunspots_births$phi))

# Test rotational symmetry for the 23rd cycle, specified theta
sunspots_23 <- subset(sunspots_births, cycle == 23)
```

```
test_rotasym(data = sunspots_23$X, type = "sc", theta = c(0, 0, 1))
test_rotasym(data = sunspots_23$X, type = "loc", theta = c(0, 0, 1))
test_rotasym(data = sunspots_23$X, type = "hyb", theta = c(0, 0, 1))

# Test rotational symmetry for the 23rd cycle, unspecified theta
spherical_loc_PCA(sunspots_23$X)
test_rotasym(data = sunspots_23$X, type = "sc", theta = spherical_loc_PCA)
test_rotasym(data = sunspots_23$X, type = "loc_vMF",
             theta = spherical_loc_PCA)
test_rotasym(data = sunspots_23$X, type = "hyb_vMF",
             theta = spherical_loc_PCA)

# Test rotational symmetry for the 22nd cycle, specified theta
sunspots_22 <- subset(sunspots_births, cycle == 22)
test_rotasym(data = sunspots_22$X, type = "sc", theta = c(0, 0, 1))
test_rotasym(data = sunspots_22$X, type = "loc", theta = c(0, 0, 1))
test_rotasym(data = sunspots_22$X, type = "hyb", theta = c(0, 0, 1))

# Test rotational symmetry for the 22nd cycle, unspecified theta
spherical_loc_PCA(sunspots_22$X)
test_rotasym(data = sunspots_22$X, type = "sc", theta = spherical_loc_PCA)
test_rotasym(data = sunspots_22$X, type = "loc_vMF",
             theta = spherical_loc_PCA)
test_rotasym(data = sunspots_22$X, type = "hyb_vMF",
             theta = spherical_loc_PCA)
```

`unif` *Uniform distribution on the hypersphere*

Description

Density and simulation of the uniform distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 1$. The density is just the inverse of the surface area of $\mathcal{S}^{p-1}$, given by

$$\omega_p := 2\pi^{p/2}/\Gamma(p/2).$$

Usage

```
d_unif_sphere(x, log = FALSE)

r_unif_sphere(n, p)

w_p(p, log = FALSE)
```

Arguments

`x`	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size `c(nx, p)` or a vector of length `p`. Normalized internally if required (with a `warning` message).
`log`	compute the logarithm of the density (or the normalizing constant)? Defaults to `FALSE`.

n	sample size, a positive integer.
p	dimension of the ambient space $\mathbb{R}^p$ that contains $\mathcal{S}^{p-1}$. A positive integer.

Details

If $p = 1$, then $\mathcal{S}^0 = \{-1, 1\}$ and the "surface area" is 2. The function `w_p` is vectorized on `p`.

Value

Depending on the function:

- `d_unif_sphere`: a vector of length `nx` or `1` with the evaluated density at `x`.
- `r_unif_sphere`: a matrix of size `c(n, p)` with the random sample.
- `w_p`: the surface area of $\mathcal{S}^{p-1}$.

See Also

vMF, ACG, tang-norm-decomp.

Examples

```
## Area of S^{p - 1}

# Areas of S^0, S^1, and S^2
w_p(p = 1:3)

# Area as a function of p
p <- 1:20
plot(p, w_p(p = p), type = "o", pch = 16, xlab = "p", ylab = "Area",
     main = expression("Surface area of " * S^{p - 1}), axes = FALSE)
box()
axis(1, at = p)
axis(2, at = seq(0, 34, by = 2))

## Simulation and density evaluation for p = 1, 2, 3

# p = 1
n <- 500
x <- r_unif_sphere(n = n, p = 1)
barplot(table(x) / n)
head(d_unif_sphere(x))

# p = 2
x <- r_unif_sphere(n = n, p = 2)
plot(x)
head(d_unif_sphere(x))

# p = 3
x <- r_unif_sphere(n = n, p = 3)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), pch = 16, xlab = "", ylab = "",
```

```
                                zlab = "", angle = 20)
head(d_unif_sphere(x))
```

---

vMF *von Mises–Fisher distribution*

Description

Density and simulation of the von Mises–Fisher (vMF) distribution on $\mathcal{S}^{p-1} := \{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\| = 1\}$, $p \geq 1$. The density at $\boldsymbol{x} \in \mathcal{S}^{p-1}$ is given by

$$c_{p,\kappa}^{\mathrm{vMF}} e^{\kappa \boldsymbol{x}'\boldsymbol{\mu}} \quad \text{with} \quad c_{p,\kappa}^{\mathrm{vMF}} := \kappa^{(p-2)/2} / ((2\pi)^{p/2} I_{(p-2)/2}(\kappa))$$

where $\boldsymbol{\mu} \in \mathcal{S}^{p-1}$ is the directional mean, $\kappa \geq 0$ is the concentration parameter about $\boldsymbol{\mu}$, and I_ν is the order-ν modified Bessel function of the first kind.

The angular function of the vMF is $g(t) := e^{\kappa t}$. The associated *cosines* density is $\tilde{g}(v) := \omega_{p-1} c_{p,\kappa}^{\mathrm{vMF}} g(v)(1-v^2)^{(p-3)/2}$, where ω_{p-1} is the surface area of $\mathcal{S}^{p-2}$.

Usage

```
d_vMF(x, mu, kappa, log = FALSE)

c_vMF(p, kappa, log = FALSE)

r_vMF(n, mu, kappa)

g_vMF(t, p, kappa, scaled = TRUE, log = FALSE)

r_g_vMF(n, p, kappa)
```

Arguments

x	locations in $\mathcal{S}^{p-1}$ to evaluate the density. Either a matrix of size c(nx, p) or a vector of length p. Normalized internally if required (with a warning message).
mu	the directional mean $\boldsymbol{\mu}$ of the vMF. A unit-norm vector of length p.
kappa	concentration parameter κ of the vMF. A nonnegative scalar. Can be a vector for c_vMF.
log	compute the logarithm of the density, the normalizing constant, or the angular function? Defaults to FALSE.
p	dimension of the ambient space $\mathbb{R}^p$ that contains $\mathcal{S}^{p-1}$. A positive integer.
n	sample size, a positive integer.
t	a vector with values in $[-1, 1]$.
scaled	scale the angular function by the von Mises–Fisher normalizing constant? Defaults to TRUE.

Details

`r_g_vMF` implements algorithm VM in Wood (1994), except for $p = 3$, where the cosines density is the truncated exponential $\propto e^{\kappa t}$ and is simulated exactly by inverse transform. `c_vMF` is vectorized on `p` and `kappa`, and is computed with `log_c_vmf`, which is stable for large `p` and `kappa`.

Value

Depending on the function:

- `d_vMF`: a vector of length `nx` or 1 with the evaluated density at `x`.
- `r_vMF`: a matrix of size `c(n, p)` with the random sample.
- `c_vMF`: the normalizing constant.
- `g_vMF`: a vector of length `length(t)` with the evaluated angular function.
- `r_g_vMF`: a matrix of size `c(n, 1)` containing simulated values from the cosines density associated to the angular function.

References

Wood, A. T. A. (1994). Simulation of the von Mises Fisher distribution. *Communications in Statistics – Simulation and Computation*, 23(1):157–164. doi:10.1080/03610919408813161

See Also

`tangent-vMF`, `unif`, `tang-norm-decomp`.

Examples

```
# Simulation and density evaluation for p = 2
mu <- c(0, 1)
kappa <- 2
n <- 1e3
x <- r_vMF(n = n, mu = mu, kappa = kappa)
dens <- d_vMF(x = x, mu = mu, kappa = kappa)
col <- viridisLite::viridis(n)
r <- runif(n, 0.95, 1.05) # Radius perturbation to improve visualization
plot(r * x, pch = 16, col = col[rank(dens)])

# Simulation and density evaluation for p = 3
mu <- c(0, 0, 1)
kappa <- 2
x <- r_vMF(n = n, mu = mu, kappa = kappa)
dens <- d_vMF(x = x, mu = mu, kappa = kappa)
scatterplot3d::scatterplot3d(x, xlim = c(-1, 1), ylim = c(-1, 1),
                             zlim = c(-1, 1), color = col[rank(dens)],
                             pch = 16, xlab = "", ylab = "", zlab = "",
                             angle = 20)

# Cosines density
g_tilde <- function(t, p, kappa) {
  exp(w_p(p = p - 1, log = TRUE) +
```

```
        g_vMF(t = t, p = p, kappa = kappa, scaled = TRUE, log = TRUE) +
        ((p - 3) / 2) * log(1 - t^2))
}

# Simulated data from the cosines density
n <- 1e3
p <- 3
kappa <- 2
hist(r_g_vMF(n = n, p = p, kappa = kappa), breaks = seq(-1, 1, l = 20),
     probability = TRUE, main = "Simulated data from g_vMF", xlab = "t")
t <- seq(-1, 1, by = 0.01)
lines(t, g_tilde(t = t, p = p, kappa = kappa))

# Cosine density as a function of the dimension
M <- 100
col <- viridisLite::viridis(M)
plot(t, g_tilde(t = t, p = 2, kappa = kappa), col = col[2], type = "l",
     ylab = "Density")
for (p in 3:M) {
  lines(t, g_tilde(t = t, p = p, kappa = kappa), col = col[p])
}
```

Index

∗ **datasets**
    sunspots, 8

ACG, 3, *16*, *26*
Angular Central Gaussian, *15*
angular functions, *7*

c_ACG *(*ACG*)*, 3
c_vMF *(*vMF*)*, 27
cosines *(*cosines-signs*)*, 4
cosines-signs, 4

d_ACG, *16*
d_ACG *(*ACG*)*, 3
d_tang_norm, *16*, *18*
d_tang_norm *(*tang-norm-decomp*)*, 12
d_TE *(*tangent-elliptical*)*, 15
d_TM *(*tangent-vMF*)*, 18
d_unif_sphere *(*unif*)*, 25
d_vMF, *18*
d_vMF *(*vMF*)*, 27

estimators, 7, *22*

g_vMF, *13*, *21*
g_vMF *(*vMF*)*, 27
Gamma_theta, *12*
Gamma_theta *(*cosines-signs*)*, 4

log_c_vmf, *28*

POSIXct, *8*
prod_Gamma_theta *(*cosines-signs*)*, 4

r_ACG, *16*
r_ACG *(*ACG*)*, 3
r_g_vMF *(*vMF*)*, 27
r_tang_norm, *16*, *18*
r_tang_norm *(*tang-norm-decomp*)*, 12
r_TE *(*tangent-elliptical*)*, 15
r_TM *(*tangent-vMF*)*, 18
r_unif_sphere *(*unif*)*, 25
r_vMF, *18*
r_vMF *(*vMF*)*, 27
rotasym *(*rotasym-package*)*, 2
rotasym-package, 2

signs, *15*, *18*
signs *(*cosines-signs*)*, 4
spherical_loc_PCA *(*estimators*)*, 7
spherical_mean, *21*
spherical_mean *(*estimators*)*, 7
sunspots, 8
sunspots_births *(*sunspots*)*, 8
sunspots_deaths *(*sunspots*)*, 8

tang-norm-decomp, 12
tangent elliptical, *21*
tangent von Mises-Fisher, *21*
tangent-elliptical, 15
tangent-normal decomposition, *15*, *18*
tangent-vMF, 18
TE *(*tangent-elliptical*)*, 15
test_rotasym, *6*, *7*, *9*, *16*, *19*, 20
TM *(*tangent-vMF*)*, 18

unif, *4*, 25, *28*

vMF, *19*, *26*, 27
von Mises-Fisher, *18*

w_p *(*unif*)*, 25