Package 'squat'

December 22, 2022

Title Statistics for Quaternion Temporal Data

Version 0.1.0

Description An implementation of statistical tools for the analysis of unit quaternion time series. It relies on pre-existing quaternion data structure provided by the 'Eigen' C++ library.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

LinkingTo Rcpp, RcppArmadillo, RcppEigen, fdacluster

Imports cli, dtw, fdacluster, funData, furrr, ggplot2, ggrepel, MFPCA, progressr, purrr, Rcpp, roahd, scales, tibble, tidyr

Depends R (>= 4.1.0)

Suggests covr, future, gganimate, gghighlight, testthat (>= 3.0.0), vdiffr, withr

Config/testthat/edition 3

URL https://lmjl-alea.github.io/squat/

NeedsCompilation yes

Author Lise Bellanger [aut], Pierre Drouin [aut], Aymeric Stamm [aut, cre] (<https://orcid.org/0000-0002-8725-3654>), Benjamin Martineau [ctb]

Maintainer Aymeric Stamm <aymeric.stamm@math.cnrs.fr>

Repository CRAN

Date/Publication 2022-12-22 11:20:02 UTC

R topics documented:

---

append *QTS Sample Concatenation*

Description

QTS Sample Concatenation

Usage

```
append(x, ...)

## Default S3 method:
append(x, values, after = length(x), ...)

## S3 method for class 'qts_sample'
append(x, y, ...)
```

Arguments

x	An object of class qts_sample.
...	Extra arguments to be passed on to next methods.
values	to be included in the modified vector.
after	a subscript, after which the values are to be appended.
y	Either an object of class qts_sample or an object of class qts.

Examples

```
append(vespa64$igp, vespa64$igp[1])
append(vespa64$igp, vespa64$igp[[1]])
```

---

centring *QTS Centering and Standardization*

Description

This function operates a centering of the QTS around the geometric mean of its quaternions. This is effectively achieved by left-multiplying each quaternion by the inverse of their geometric mean.

Usage

```
centring(x, standardize = FALSE, keep_summary_stats = FALSE)
```

Arguments

x	An object of class qts.
standardize	A boolean specifying whether to standardize the QTS in addition to centering it. Defaults to FALSE.
keep_summary_stats	A boolean specifying whether the mean and standard deviation used for standardizing the data should be stored in the output object. Defaults to FALSE in which case only the centered qts is returned.

Value

If keep_summary_stats = FALSE, an object of class qts in which quaternions have been centered (and possibly standardized) around their geometric mean. If keep_summary_stats = TRUE, a list with three components:

- qts: an object of class qts in which quaternions have been centered (and possibly standardized) around their geometric mean;
- mean: a numeric vector with the quaternion Fréchet mean;
- sd: a numeric value with the quaternion Fréchet standard deviation.

Examples

```
centring(vespa64$igp[[1]])
```

differentiate *QTS Differentiation*

Description

This function computes the first derivative of quaternion time series with respect to time.

Usage

```
differentiate(x)

## S3 method for class 'qts'
differentiate(x)

## S3 method for class 'qts_sample'
differentiate(x)
```

Arguments

x	An object of class qts or qts_sample.

Value

An object of the same class as the input argument x in which quaternions measure the rotation to be applied to transform attitude at previous time point to attitude at current time point.

Examples

```
differentiate(vespa64$igp[[1]])
differentiate(vespa64$igp)
```

distDTW *Distance Matrix for Quaternion Time Series Samples*

Description

Distance Matrix for Quaternion Time Series Samples

Usage

```
distDTW(
  qts_list,
  normalize_distance = TRUE,
  labels = NULL,
  resample = TRUE,
  disable_normalization = FALSE,
  step_pattern = dtw::symmetric2
)
```

Arguments

qts_list	An object of class qts_sample.
normalize_distance	A boolean specifying whether to compute normalized distance between QTS. Please note that not all step patterns are normalizable. Defaults to FALSE.
labels	A character vector specifying labels for each QTS. Defaults to NULL which uses row numbers as labels.
resample	A boolean specifying whether the QTS should be uniformly resampled on their domain before computing distances. Defaults to TRUE.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE which ensures that we always deal with unit quaternions.
step_pattern	A dtw::stepPattern specifying the local constraints on the warping path. Defaults to dtw::symmetric2 which uses symmetric and normalizable warping paths with no local slope constraints. See dtw::stepPattern for more information.

Value

A stats::dist object storing the distance matrix between QTS in a sample via DTW.

Examples

```
D <- distDTW(vespa64$igp)
```

DTW *Dynamic Time Warping for Quaternion Time Series*

Description

This function evaluates the Dynamic Time Warping (DTW) distance between two quaternion time series (QTS).

Usage

```
DTW(
  qts1,
  qts2,
  resample = TRUE,
  disable_normalization = FALSE,
  distance_only = FALSE,
  step_pattern = dtw::symmetric2
)
```

Arguments

qts1	An object of class qts.
qts2	An object of class qts.
resample	A boolean specifying whether the QTS should be uniformly resampled on their domain before computing distances. Defaults to TRUE.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE which ensures that we always deal with unit quaternions.
distance_only	A boolean specifying whether to only compute distance (no backtrack, faster). Defaults to FALSE.
step_pattern	A dtw::stepPattern specifying the local constraints on the warping path. Defaults to dtw::symmetric2 which uses symmetric and normalizable warping paths with no local slope constraints. See dtw::stepPattern for more information.

Details

If no evaluation grid is provided, the function assumes that the two input QTS are evaluated on the same grid.

Value

An object of class dtw::dtw storing the dynamic time warping results.

Examples

```
DTW(vespa64$igp[[1]], vespa64$igp[[2]])
```

---

exp *QTS Exponential*

Description

This function computes the exponential of quaternion time series as the time series of the quaternion exponentials.

Usage

```
## S3 method for class 'qts'
exp(x, ...)

## S3 method for class 'qts_sample'
exp(x, ...)
```

Arguments

x	An object of class qts or qts_sample.
...	Extra arguments to be passed on to next methods.

Value

An object of the same class as the input argument x in which quaternions have been replaced by their exponential.

Examples

```
x <- log(vespa64$igp[[1]])
exp(x)
y <- log(vespa64$igp)
exp(y)
```

`hemispherize` *QTS Hemispherization*

Description

This function ensures that there are no discontinuities in QTS due to quaternion flips since two unit quaternions q and -q encode the same rotation.

Usage

```
hemispherize(x)

## S3 method for class 'qts'
hemispherize(x)

## S3 method for class 'qts_sample'
hemispherize(x)
```

Arguments

x	An object of class `qts` or `qts_sample`.

Value

An object of the same class as the input argument x with no quaternion flip discontinuities.

Examples

```
hemispherize(vespa64$igp[[1]])
hemispherize(vespa64$igp)
```

kmeans *QTS K-Means Alignment Algorithm*

Description

This function massages the input quaternion time series to feed them into the k-means alignment algorithm for jointly clustering and aligning the input QTS.

Usage

```
kmeans(x, k, iter_max = 10, nstart = 1, ...)

## Default S3 method:
kmeans(
  x,
  k,
  iter_max = 10,
  nstart = 1,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  trace = FALSE,
  ...
)

## S3 method for class 'qts_sample'
kmeans(
  x,
  k = 1,
  iter_max = 10,
  nstart = 1,
  centroid = "mean",
  dissimilarity = "l2",
  warping = "affine",
  ...
)
```

Arguments

x	Either a numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns) or an object of class qts_sample.
k	An integer value specifying the number of clusters to be look for.
iter_max	An integer value specifying the maximum number of iterations for terminating the k-mean algorithm. Defaults to 10L.
nstart	An integer value specifying the number of random restarts of the algorithm. The higher nstart, the more robust the result. Defaults to 1L.
...	not used.

algorithm	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm.
trace	logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
centroid	A string specifying which type of centroid should be used when applying kmeans on a QTS sample. Choices are mean and medoid. Defaults to mean.
dissimilarity	A string specifying which type of dissimilarity should be used when applying kmeans on a QTS sample. Choices are l2 and pearson. Defaults to l2.
warping	A string specifying which class of warping functions should be used when applying kmeans on a QTS sample. Choices are none, shift, dilation and affine. Defaults to affine.

Value

An object of class stats::kmeans if the input x is NOT of class qts_sample. Otherwise, an object of class kma_qts which is effectively a list with three components:

- qts_aligned: An object of class qts_sample storing the sample of aligned QTS;
- qts_centers: A list of objects of class qts representing the centers of the clusters;
- best_kma_result: An object of class fdacluster::kma storing the results of the best k-mean alignment result among all initialization that were tried.

Examples

```
res_kma <- kmeans(vespa64$igp, k = 2)
```

---

log *QTS Logarithm*

Description

This function computes the logarithm of quaternion time series as the time series of the quaternion logarithms.

Usage

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

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

Arguments

x	An object of class qts or qts_sample.
...	Extra arguments to be passed on to next methods.

Value

An object of the same class as the input argument x in which quaternions have been replaced by their logarithm.

Examples

```
log(vespa64$igp[[1]])
log(vespa64$igp)
```

`mean.qts_sample` *QTS Geometric Mean*

Description

This function computes the pointwise geometric mean of a QTS sample.

Usage

```
## S3 method for class 'qts_sample'
mean(x, ...)
```

Arguments

x	An object of class qts_sample.
...	Further arguments passed to or from other methods.

Value

An object of class qts in which quaternions are the pointwise geometric mean of the input QTS sample.

Examples

```
mean(vespa64$igp)
```

median.qts_sample *QTS Geometric Median*

Description

This function computes the pointwise geometric median of a QTS sample.

Usage

```
## S3 method for class 'qts_sample'
median(x, na.rm = FALSE, ...)
```

Arguments

x	An object of class qts_sample.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

An object of class qts in which quaternions are the pointwise geometric median of the input QTS sample.

Examples

```
median(vespa64$igp)
```

moving_average *QTS Moving Average*

Description

This function performs QTS smoothing via moving average.

Usage

```
moving_average(x, window_size = 0)

## S3 method for class 'qts'
moving_average(x, window_size = 0)

## S3 method for class 'qts_sample'
moving_average(x, window_size = 0)
```

Arguments

x	An object of class qts or qts_sample.
window_size	An integer value specifying the size of the sliding window used to compute the median value. Defaults to 0L.

Value

An object of the same class as the input argument x storing the smoothed QTS.

Examples

```
moving_average(vespa64$igp[[1]], window_size = 5)
moving_average(vespa64$igp, window_size = 5)
```

---

normalize *QTS Normalization*

Description

This function ensures that all quaternions in the time series are unit quaternions.

Usage

```
normalize(x)

## S3 method for class 'qts'
normalize(x)

## S3 method for class 'qts_sample'
normalize(x)
```

Arguments

x	An object of class qts or qts_sample.

Value

An object of the same class as the input argument x in which quaternions are unit quaternions.

Examples

```
normalize(vespa64$igp[[1]])
normalize(vespa64$igp)
```

plot.kma_qts *QTS K-Means Visualization*

Description

QTS K-Means Visualization

Usage

```
## S3 method for class 'kma_qts'
plot(x, ...)

## S3 method for class 'kma_qts'
autoplot(x, ...)
```

Arguments

x	An object of class kma_qts as produced by the kmeans() function.
...	Further arguments to be passed to other methods.

Value

The plot.kma_qts() method does not return anything while the autoplot.kma_qts() method returns a ggplot2::ggplot object.

Examples

```
res_kma <- kmeans(vespa64$igp, k = 2, nstart = 1)
plot(res_kma)
ggplot2::autoplot(res_kma)
```

plot.prcomp_qts *QTS PCA Visualization*

Description

QTS PCA Visualization

Usage

```
## S3 method for class 'prcomp_qts'
plot(x, what = "PC1", ...)

## S3 method for class 'prcomp_qts'
autoplot(x, what = "PC1", ...)

## S3 method for class 'prcomp_qts'
screeplot(x, ...)
```

Arguments

x	An object of class prcomp_qts as produced by the prcomp.qts_sample() method.
what	A string specifying what kind of visualization the user wants to perform. Choices are words starting with PC and ending with a PC number (in which case the mean QTS is displayed along with its perturbations due to the required PC) or scores (in which case individuals are projected on the required plane). Defaults to PC1.
...	If what = "PC?", the user can specify whether to plot the QTS in the tangent space or in the original space by providing a boolean argument original_space which defaults to TRUE. If what = "scores", the user can specify the plane onto which the individuals will be projected by providing a length-2 integer vector argument plane which defaults to 1:2.

Value

The plot.prcomp_qts() method does not return anything while the autoplot.prcomp_qts() method returns a ggplot2::ggplot object.

Examples

```
df <- as_qts_sample(vespa64$igp[1:16])
res_pca <- prcomp(df)

# You can plot the effect of a PC on the mean
plot(res_pca, what = "PC1")

# You can plot the data points in a PC plane
plot(res_pca, what = "scores")

# You can color points according to a categorical variable
if (requireNamespace("ggplot2", quietly = TRUE)) {
  p <- ggplot2::autoplot(res_pca, what = "scores")
  p + ggplot2::geom_point(ggplot2::aes(color = vespa64$V[1:16]))
}
```

---

plot.qts *QTS Visualization*

Description

QTS Visualization

Usage

```
## S3 method for class 'qts'
plot(x, highlighted_points = NULL, ...)

## S3 method for class 'qts'
autoplot(x, highlighted_points = NULL, ...)
```

Arguments

x	An object of class qts.
highlighted_points	An integer vector specifying point indices to be highlighted. Defaults to NULL, in which case no point will be highlighted with respect to the others.
...	Further arguments to be passed on to next methods.

Value

The plot.qts() method does not return anything while the autoplot.qts() method returns a ggplot2::ggplot object.

Examples

```
plot(vespa64$igp[[1]])
ggplot2::autoplot(vespa64$igp[[1]])
```

---

plot.qts_sample *QTS Sample Visualization*

Description

QTS Sample Visualization

Usage

```
## S3 method for class 'qts_sample'
plot(x, memberships = NULL, highlighted = NULL, with_animation = FALSE, ...)

## S3 method for class 'qts_sample'
autoplot(
  x,
  memberships = NULL,
  highlighted = NULL,
  with_animation = FALSE,
  ...
)
```

Arguments

x	An object of class qts_sample.
memberships	A vector coercible as factor specifying a group membership for each QTS in the sample. Defaults to NULL, in which case no grouping structure is displayed.
highlighted	A boolean vector specifying whether each QTS in the sample should be hightlighted. Defaults to NULL, in which case no QTS is hightlighted w.r.t. the others.
with_animation	A boolean value specifying whether to create a an animated plot or a static ggplot2::ggplot object. Defaults to FALSE which will create a static plot.
...	Further arguments to be passed to methods.

Value

The plot.qts_sample() method does not return anything while the autoplot.qts_sample() method returns a ggplot2::ggplot object.

Examples

```
plot(vespa64$igp)
ggplot2::autoplot(vespa64$igp)
```

---

prcomp.qts_sample *PCA for QTS Sample*

Description

PCA for QTS Sample

Usage

```
## S3 method for class 'qts_sample'
prcomp(x, M = 5, fit = FALSE, ...)
```

Arguments

x	An object of class qts_sample.
M	An integer value specifying the number of principal component to compute. Defaults to 5L.
fit	A boolean specifying whether the resulting prcomp_qts object should store a reconstruction of the sample from the retained PCs. Defaults to FALSE.
...	Arguments passed to or from other methods.

Value

An object of class prcomp_qts which is a list with the following components:

- tpca: An object of class MFPCAfit as produced by the function MFPCA::MFPCA(),
- var_props: A numeric vector storing the percentage of variance explained by each PC,
- mean_qts: An object of class qts containing the mean QTS,
- principal_qts: A list of qtss containing the required principal components.

Examples

```
res_pca <- prcomp(vespa64$igp)
```

qts *QTS Class*

Description

A collection of functions that implements the QTS class. It currently provides the `as_qts()` function for QTS coercion of `tibble::tibble`s and the `is_qts()` function for checking if an object is a QTS.

Usage

```
as_qts(x)

is_qts(x)

## S3 method for class 'qts'
format(x, digits = 5, ...)
```

Arguments

`x`	A `tibble::tibble` with columns `time`, `w`, `x`, `y` and `z`.
`digits`	An integer value specifying the number of digits to keep for printing. Defaults to `5L`.
`...`	Further arguments passed to or from other methods.

Details

A quaternion time series (QTS) is stored as a `tibble::tibble` with 5 columns:

- `time`: A first column specifying the time points at which quaternions were collected;
- `w`: A second column specifying the first coordinate of the collected quaternions;
- `x`: A third column specifying the second coordinate of the collected quaternions;
- `y`: A fourth column specifying the third coordinate of the collected quaternions;
- `z`: A fifth column specifying the fourth coordinate of the collected quaternions.

Value

An object of class qts.

Examples

```
qts1 <- vespa64$igp[[1]]
qts2 <- as_qts(qts1)
is_qts(qts1)
is_qts(qts2)
```

qts2ats *QTS Transformation To Angle Time Series*

Description

This function computes a univariate time series representing the angle between the first and other attitudes.

Usage

```
qts2ats(x, disable_normalization = FALSE)
```

Arguments

x	An object of class qts.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE.

Value

A time series stored as a tibble::tibble with columns time and angle in which angle measures the angle between the current rotation and the first one.

Examples

```
qts2ats(vespa64$igp[[1]])
```

qts2avts *QTS Transformation to Angular Velocity Time Series*

Description

This function projects a quaternion time series into the space of angular velocities.

Usage

```
qts2avts(x, body_frame = FALSE)
```

Arguments

x	An object of class qts.
body_frame	A boolean specifying whether the fixed frame with respect to which coordinates of the angular velocity should be computed is the body frame or the global frame. Defaults to FALSE.

Value

A time series stored as a tibble::tibble with columns `time`, `x`, `y` and `z` containing the angular velocity at each time point.

Examples

```
qts2avts(vespa64$igp[[1]])
```

`qts2dts` *QTS Transformation To Distance Time Series*

Description

This function computes a real-valued time series reporting the pointwise geodesic distance between the two input QTS at each time point.

Usage

```
qts2dts(x, y)
```

Arguments

`x`	An object of class qts.
`y`	An object of class qts.

Details

The function currently expects that the two input QTS are evaluated on the same time grid.

Value

A time series stored as a tibble::tibble with columns `time` and `distance` in which `distance` measures the angular distance between the quaternions of both input QTS at a given time point.

Examples

```
qts2dts(vespa64$igp[[1]], vespa64$igp[[2]])
```

qts2nts *QTS Transformation To Norm Time Series*

Description

This function computes a univariate time series representing the norm of the quaternions.

Usage

```
qts2nts(x, disable_normalization = FALSE)
```

Arguments

x	An object of class qts.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE.

Value

A time series stored as a tibble::tibble with columns time and norm in which norm measures the angular distance between the current quaternion and the identity.

Examples

```
qts2nts(vespa64$igp[[1]])
```

qts_sample *QTS Sample Class*

Description

A collection of functions that implements the QTS sample class. It currently provides the as_qts_sample() function for QTS sample coercion of lists of qts objects, the is_qts_sample() function for checking if an object is a QTS sample and the subset operator.

Usage

```
as_qts_sample(x)

is_qts_sample(x)

## S3 method for class 'qts_sample'
x[i, simplify = FALSE]
```

Arguments

x	A list of tibble::tibbles, each of which with columns time, w, x, y and z.
i	A valid expression to subset observations from a QTS sample.
simplify	A boolean value specifying whether the resulting subset should be turned into a single QTS in case the subset is of size 1. Defaults to FALSE.

Details

A QTS sample is a collection of quaternion time series (QTS), each of which is stored as a tibble::tibble with 5 columns:

- time: A first column specifying the time points at which quaternions were collected;
- w: A second column specifying the first coordinate of the collected quaternions;
- x: A third column specifying the second coordinate of the collected quaternions;
- y: A fourth column specifying the third coordinate of the collected quaternions;
- z: A fifth column specifying the fourth coordinate of the collected quaternions.

Value

An object of class qts_sample.

Examples

```
x <- vespa64$igp
y <- as_qts_sample(x)
is_qts_sample(x)
is_qts_sample(y)
x[1]
x[1, simplify = TRUE]
```

---

reorient *QTS Reorientation*

Description

This function reorients the quaternions in a QTS for representing attitude with respect to the orientation of the sensor at the first time point.

Usage

```
reorient(x, disable_normalization = FALSE)

## S3 method for class 'qts'
reorient(x, disable_normalization = FALSE)

## S3 method for class 'qts_sample'
reorient(x, disable_normalization = FALSE)
```

Arguments

x	An object of class qts or qts_sample.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE.

Value

An object of the same class as the input argument x in which quaternions measure attitude with respect to the orientation of the sensor at the first time point.

Examples

```
reorient(vespa64$igp[[1]])
reorient(vespa64$igp)
```

---

resample *QTS Resampling*

Description

This function performs uniform resampling using SLERP.

Usage

```
resample(x, tmin = NA, tmax = NA, nout = 0L, disable_normalization = FALSE)

## S3 method for class 'qts'
resample(x, tmin = NA, tmax = NA, nout = 0L, disable_normalization = FALSE)

## S3 method for class 'qts_sample'
resample(x, tmin = NA, tmax = NA, nout = 0L, disable_normalization = FALSE)
```

Arguments

x	An object of class qts or qts_sample.
tmin	A numeric value specifying the lower bound of the time interval over which uniform resampling should take place. It must satisfy tmin >= min(qts$time). Defaults to NA in which case it is set to min(qts$time).
tmax	A numeric value specifying the upper bound of the time interval over which uniform resampling should take place. It must satisfy tmax <= max(qts$time). Defaults to NA in which case it is set to max(qts$time).
nout	An integer specifying the size of the uniform grid for time resampling. Defaults to 0L in which case it uses the same grid size as the input QTS.
disable_normalization	A boolean specifying whether quaternion normalization should be disabled. Defaults to FALSE in which case the function makes sure that quaternions are normalized prior to performing SLERP interpolation.

Value

An object of the same class as the input argument `x` in which quaternions are uniformly sampled in the range `[tmin, tmax]`.

Examples

```
resample(vespa64$igp[[1]])
resample(vespa64$igp)
```

`rnorm_qts` *QTS Random Sampling*

Description

This function adds uncorrelated Gaussian noise to the logarithm QTS using an exponential covariance function.

Usage

```
rnorm_qts(n, mean_qts, alpha = 0.01, beta = 0.001)
```

Arguments

`n`	An integer specifying how many QTS should be generated.
`mean_qts`	An object of class `qts` specifying the mean QTS.
`alpha`	A positive scalar specifying the variance of each component of the log-QTS. Defaults to `0.01`.
`beta`	A positive scalar specifying the exponential weight. Defaults to `0.001`.

Details

See `exp_cov_function` for details about the roles of `alpha` and `beta` in the definition of the covariance operator.

Value

A list of `n` objects of class `qts` with added noise as specified by parameters `alpha` and `beta`.

Examples

```
rnorm_qts(1, vespa64$igp[[1]])
```

scale *QTS Sample Centering and Standardization*

Description

QTS Sample Centering and Standardization

Usage

```
scale(x, center = TRUE, scale = TRUE, ...)

## Default S3 method:
scale(x, center = TRUE, scale = TRUE, ...)

## S3 method for class 'qts_sample'
scale(
  x,
  center = TRUE,
  scale = TRUE,
  by_row = FALSE,
  keep_summary_stats = FALSE,
  ...
)
```

Arguments

x	An object coercible into a numeric matrix or an object of class qts_sample representing a sample of observed QTS.
center	A boolean specifying whether to center the sample. If set to FALSE, the original sample is returned, meaning that no standardization is performed regardless of whether argument scale was set to TRUE or not. Defaults to TRUE.
scale	A boolean specifying whether to standardize the sample once it has been centered. Defaults to TRUE.
...	Extra arguments passed on to next methods.
by_row	A boolean specifying whether the QTS scaling should happen for each data point (by_row = TRUE) or for each time point (by_row = FALSE). Defaults to FALSE.
keep_summary_stats	A boolean specifying whether the mean and standard deviation used for standardizing the data should be stored in the output object. Defaults to FALSE in which case only the list of properly rescaled QTS is returned.

Value

A list of properly rescaled QTS stored as an object of class qts_sample when keep_summary_stats = FALSE. Otherwise a list with three components:

- `rescaled_sample`: a list of properly rescaled QTS stored as an object of class `qts_sample`;
- `mean`: a list of numeric vectors storing the corresponding quaternion Fréchet means;
- `sd`: a numeric vector storing the corresponding quaternion Fréchet standard deviations.

Examples

```
x <- scale(vespa64$igp)
x[[1]]
```

`smooth` *QTS Smoothing via SLERP Interpolation*

Description

This function performs a smoothing of a QTS by SLERP interpolation.

Usage

```
smooth(x, ...)

## Default S3 method:
smooth(
  x,
  kind = c("3RS3R", "3RSS", "3RSR", "3R", "3", "S"),
  twiceit = FALSE,
  endrule = c("Tukey", "copy"),
  do.ends = FALSE,
  ...
)

## S3 method for class 'qts'
smooth(x, alpha = 0.5, ...)

## S3 method for class 'qts_sample'
smooth(x, alpha = 0.5, ...)
```

Arguments

`x`	An object of class qts or qts_sample.
`...`	Extra arguments passed on to next methods.
`kind`	a character string indicating the kind of smoother required; defaults to `"3RS3R"`.
`twiceit`	logical, indicating if the result should be 'twiced'. Twicing a smoother $S(y)$ means $S(y) + S(y - S(y))$, i.e., adding smoothed residuals to the smoothed values. This decreases bias (increasing variance).
`endrule`	a character string indicating the rule for smoothing at the boundary. Either `"Tukey"` (default) or `"copy"`.

do.ends	logical, indicating if the 3-splitting of ties should also happen at the boundaries (ends). This is only used for kind = "S".
alpha	A numeric value in [0,1] specifying the amount of smoothing. The closer to one, the smoother the resulting QTS. Defaults to 0.5.

Value

An object of the same class as the input argument x which is a smooth version of the input QTS.

Examples

```
smooth(vespa64$igp[[1]])
smooth(vespa64$igp)
```

---

straighten *QTS Straightening*

Description

This function straightens QTS so that the last point equals the first point.

Usage

```
straighten(x)

## S3 method for class 'qts'
straighten(x)

## S3 method for class 'qts_sample'
straighten(x)
```

Arguments

x	An object of class qts or qts_sample.

Value

An object of the same class as the input argument x storing the straightened QTS.

Examples

```
straighten(vespa64$igp[[1]])
straighten(vespa64$igp)
```

vespa *The VESPA dataset*

Description

A set of QTS representing individual gait patterns (IGPs) of individuals collected under a number of varying factors.

Usage

```
vespa
```

Format

A `tibble` with 320 rows and 7 columns:

- `V`: a categorical variable with two levels specifying the ID of the `Volunteer`;
- `E`: a categorical variable with two levels specifying the ID of the `Experimenter`;
- `S`: a categorical variable with four levels specifying the type of `Sensor`;
- `P`: a categorical variable with four levels specifying the `Position` of the sensor;
- `A`: a categorical variable with two levels specifying the ID of the `Acquisition` pathway;
- `R`: a categorical variable with 5 levels specifying the ID of the `Repetition`;
- `igp`: A 101x5 `tibble` storing a QTS which represents the IGP of the individual under a specific set of VESPA conditions.

Details

The IGP measures the hip rotation during a typical gait cycle. Each rotation is expressed with respect to the mean position of the sensor during the gait cycle. Each IGP is then straightened so that it is periodic with a last point matching the first one.

vespa64 *The VESPA64 dataset*

Description

A set of QTS representing individual gait patterns (IGPs) of individuals collected under a number of varying factors.

Usage

```
vespa64
```

Format

A tibble with 320 rows and 7 columns:

- V: a categorical variable with two levels specifying the ID of the Volunteer;
- E: a categorical variable with two levels specifying the ID of the Experimenter;
- S: a categorical variable with four levels specifying the type of Sensor;
- P: a categorical variable with four levels specifying the Position of the sensor;
- A: a categorical variable with two levels specifying the ID of the Acquisition pathway;
- igp: A 101x5 tibble storing a QTS which represents the IGP of the individual under a specific set of VESPA conditions.

Details

The IGP measures the hip rotation during a typical gait cycle. Each rotation is expressed with respect to the mean position of the sensor during the gait cycle. Each IGP is then straightened so that it is periodic with a last point matching the first one.

It is essentially a reduced version of the VESPA data set where IGPs have been averaged over the repetition for each set of conditions.

Index