Package ‘weird’

January 24, 2024

Title Functions and Data Sets for ``That's Weird: Anomaly Detection Using R'' by Rob J Hyndman

Version 1.0.2

Description All functions and data sets required for the examples in the book Hyndman (2024) ``That's Weird: Anomaly Detection Using R'' <https://OTexts.com/weird/>. All packages needed to run the examples are also loaded.

Imports aplpack, broom, cli (>= 1.0.0), crayon (>= 1.3.4), dbscan, dplyr (>= 0.7.4), evd, ggplot2 (>= 3.1.1), grDevices, interpolation, ks, purrr (>= 0.2.4), rlang, robustbase, rstudioapi (>= 0.7), stray, tibble (>= 1.4.2)

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.0

Depends R (>= 4.1.0)

Suggests mgcv, outliers, testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

URL https://pkg.robjhyndman.com/weird-package/, https://github.com/robjhyndman/weird-package

BugReports https://github.com/robjhyndman/weird-package/issues

NeedsCompilation no

Author Rob Hyndman [aut, cre, cph] (<https://orcid.org/0000-0002-2140-5352>), RStudio [cph]

Maintainer Rob Hyndman <Rob.Hyndman@monash.edu>

Repository CRAN

Date/Publication 2024-01-24 14:50:02 UTC

R topics documented:

`as_kde` *Convert data frame or matrix object to kde class*

Description

A density specified as a data frame or matrix can be converted to a kde object. This is useful for plotting the density using `autoplot.kde`. As kde objects are defined on a grid, the density values are interpolated based on the points in the data frame or matrix.

Usage

```
as_kde(object, density_column, ngrid, ...)
```

Arguments

`object`	Data frame or matrix with numerical columns, where one column (specified by `density_column`) contains the density values, and the remaining columns define the points at which the density is evaluated.
`density_column`	Name of the column containing the density values, specified as a bare expression. If missing, the last column is used.

`ngrid`	Number of points to use for the grid in each dimension. Default is 10001 for univariate densities and 101 for multivariate densities.
`...`	Additional arguments are ignored.

Value

An object of class "kde"

Author(s)

Rob J Hyndman

Examples

```
tibble(y = seq(-4, 4, by = 0.01), density = dnorm(y)) |>
  as_kde()
```

`autoplot.kde` *Produce ggplot of densities in 1 or 2 dimensions*

Description

Produce ggplot of densities in 1 or 2 dimensions

Usage

```
## S3 method for class 'kde'
autoplot(
  object,
  prob = seq(9)/10,
  fill = FALSE,
  show_hdr = FALSE,
  show_points = FALSE,
  show_mode = FALSE,
  show_lookout = FALSE,
  color = "#00659e",
  palette = hdr_palette,
  alpha = ifelse(fill, 1, min(1, 1000/NROW(object$x))),
  ...
)
```

Arguments

`object`	Probability density function as estimated by `ks::kde()`.
`prob`	Probability of the HDR contours to be drawn (for a bivariate plot only).
`fill`	If `TRUE`, and the density is bivariate, the bivariate contours are shown as filled regions rather than lines.

`show_hdr`	If `TRUE`, and the density is univariate, then the HDR regions specified by `prob` are shown as a ribbon below the density.
`show_points`	If `TRUE`, then individual points are plotted.
`show_mode`	If `TRUE`, then the mode of the distribution is shown.
`show_lookout`	If `TRUE`, then the observations with lookout probabilities less than 0.05 are shown in red.
`color`	Color used for mode and HDR contours. If `palette = hdr_palette`, it is also used as the basis for HDR regions.
`palette`	Color palette function to use for HDR filled regions (if `fill` is `TRUE` or `show_hdr` is `TRUE`).
`alpha`	Transparency of points. When `fill` is `FALSE`, defaults to min(1, 1000/n), where n is the number of observations. Otherwise, set to 1.
`...`	Additional arguments are currently ignored.

Details

This function produces a ggplot of the density estimate produced by `ks::kde()`. For univariate densities, it produces a line plot of the density function, with an optional ribbon showing some highest density regions (HDRs) and/or the observations. For bivariate densities, it produces a contour plot of the density function, with the observations optionally shown as points. The mode can also be drawn as a point with the HDRs. For bivariate densities, the combination of `fill = TRUE`, `show_points = TRUE`, `show_mode = TRUE`, and `prob = c(0.5, 0.99)` is equivalent to an HDR boxplot. For univariate densities, the combination of `show_hdr = TRUE`, `show_points = TRUE`, `show_mode = TRUE`, and `prob = c(0.5, 0.99)` is equivalent to an HDR boxplot.

Value

A ggplot object.

Author(s)

Rob J Hyndman

Examples

```
# Univariate density
c(rnorm(500), rnorm(500, 4, 1.5)) |>
  kde() |>
  autoplot(show_hdr = TRUE, prob= c(0.5, 0.95), color = "#c14b14")
ymat <- tibble(y1 = rnorm(5000), y2 = y1 + rnorm(5000))
ymat |>
  kde(H = kde_bandwidth(ymat)) |>
  autoplot(show_points = TRUE, alpha = 0.1, fill = TRUE)
```

`cricket_batting` *Cricket batting data for international test players*

Description

A dataset containing career batting statistics for all international test players (men and women) up to 6 October 2021.

Usage

```
cricket_batting
```

Format

A data frame with 3754 rows and 15 variables:

Player Player name in form of "initials surname"
Country Country played for
Start First year of test playing career
End Last year of test playing career
Matches Number of matches played
Innings Number of innings batted
NotOuts Number of times not out
Runs Total runs scored
HighScore Highest score in an innings
HighScoreNotOut Was highest score not out?
Average Batting average at end of career
Hundreds Total number of 100s scored
Fifties Total number of 50s scored
Ducks Total number of 0s scored
Gender "Men" or "Women"

Value

Data frame

Source

https://www.espncricinfo.com

Examples

```
cricket_batting |>
  filter(Innings > 20) |>
  select(Player, Country, Matches, Runs, Average, Hundreds, Fifties, Ducks) |>
  arrange(desc(Average))
```

density_scores *Density scores*

Description

Compute density scores or leave-one-out density scores from a model or a kernel density estimate of a data set. The density scores are defined as minus the log of the conditional density, or kernel density estimate, at each observation. The leave-one-out density scores (or LOO density scores) are obtained by estimating the conditional density or kernel density estimate using all other observations.

Usage

```
density_scores(object, loo = FALSE, ...)

## Default S3 method:
density_scores(
  object,
  loo = FALSE,
  h = kde_bandwidth(object, method = "double"),
  H = kde_bandwidth(object, method = "double"),
  ...
)

## S3 method for class 'kde'
density_scores(object, loo = FALSE, ...)

## S3 method for class 'lm'
density_scores(object, loo = FALSE, ...)

## S3 method for class 'gam'
density_scores(object, loo = FALSE, ...)
```

Arguments

`object`	A model object or a numerical data set.
`loo`	Should leave-one-out density scores be computed?
`...`	Other arguments are ignored.
`h`	Bandwidth for univariate kernel density estimate. Default is `kde_bandwidth`.
`H`	Bandwidth for multivariate kernel density estimate. Default is `kde_bandwidth`.

Details

If the first argument is a numerical vector or matrix, then a kernel density estimate is computed, using a Gaussian kernel, with default bandwidth given by a robust normal reference rule. Otherwise the model is used to compute the conditional density function at each observation, from which the density scores (or possibly the LOO density scores) are obtained.

Value

A numerical vector containing either the density scores, or the LOO density scores.

Author(s)

Rob J Hyndman

See Also

`kde_bandwidth` `kde`

Examples

```
# Density scores computed from bivariate data set
of <- oldfaithful |>
  filter(duration < 7000, waiting < 7000) |>
  mutate(
    fscores = density_scores(cbind(duration, waiting)),
    loo_fscores = density_scores(cbind(duration, waiting), loo = TRUE),
    lookout_prob = lookout(density_scores = fscores, loo_scores = loo_fscores)
  )
of |>
  ggplot(aes(x = duration, y = waiting, color = lookout_prob < 0.01)) +
  geom_point()
# Density scores computed from bivariate KDE
f_kde <- kde(of[, 2:3], H = kde_bandwidth(of[, 2:3]))
of |>
  mutate(
    fscores = density_scores(f_kde),
    loo_fscores = density_scores(f_kde, loo = TRUE)
  )
# Density scores computed from linear model
of <- oldfaithful |>
  filter(duration < 7200, waiting < 7200)
lm_of <- lm(waiting ~ duration, data = of)
of |>
  mutate(
    fscore = density_scores(lm_of),
    loo_fscore = density_scores(lm_of, loo = TRUE),
    lookout_prob = lookout(density_scores = fscore, loo_scores = loo_fscore)
  ) |>
  ggplot(aes(x = duration, y = waiting, color = lookout_prob < 0.02)) +
  geom_point()
# Density scores computed from GAM
of <- oldfaithful |>
  filter(duration > 1, duration < 7200, waiting < 7200)
gam_of <- mgcv::gam(waiting ~ s(duration), data = of)
of |>
  mutate(
    fscore = density_scores(gam_of),
    lookout_prob = lookout(density_scores = fscore)
  ) |>
  filter(lookout_prob < 0.02)
```

`fetch_wine_reviews` *Wine prices and points*

Description

A data set containing data on wines from 44 countries, taken from *Wine Enthusiast Magazine* during the week of 15 June 2017. The data are downloaded and returned.

Usage

```
fetch_wine_reviews()
```

Format

A data frame with 110,203 rows and 8 columns:

country Country of origin

state State or province of origin

region Region of origin

winery Name of vineyard that made the wine

variety Variety of grape

points Points allocated by WineEnthusiast reviewer on a scale of 0-100

price Price of a bottle of wine in $US

year Year of wine extracted from `title`

Value

Data frame

Source

https://kaggle.com

Examples

```
## Not run:
wine_reviews <- fetch_wine_reviews()
wine_reviews |>
 ggplot(aes(x = points, y = price)) +
 geom_jitter(height = 0, width = 0.2, alpha = 0.1) +
 scale_y_log10()

## End(Not run)
```

gg_bagplot *Bagplot*

Description

Produces a bivariate bagplot. A bagplot is analagous to a univariate boxplot, except it is in two dimensions. Like a boxplot, it shows the median, a region containing 50% of the observations, a region showing the remaining observations other than outliers, and any outliers.

Usage

```
gg_bagplot(
  data,
  var1,
  var2,
  col = c(hdr_palette(color = "#00659e", prob = c(0.5, 0.99)), "#000000"),
  scatterplot = FALSE,
  ...
)
```

Arguments

`data`	A data frame or matrix containing the data.
`var1`	The name of the first variable to plot (a bare expression).
`var2`	The name of the second variable to plot (a bare expression).
`col`	The colors to use in the order: median, bag, loop and outliers.
`scatterplot`	A logical argument indicating if a regular bagplot is required (`FALSE`), or if a scatterplot in the same colors is required (`TRUE`).
`...`	Other arguments are passed to the `compute.bagplot` function.

Value

A ggplot object showing a bagplot or scatterplot of the data.

Author(s)

Rob J Hyndman

References

Rousseeuw, P. J., Ruts, I., & Tukey, J. W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, **52**(4), 382–387.

See Also

`bagplot`

Examples

```
gg_bagplot(n01, v1, v2)
gg_bagplot(n01, v1, v2, scatterplot = TRUE)
```

---

gg_hdrboxplot *HDR plot*

Description

Produces a 1d or 2d box plot of HDR regions. The darker regions contain observations with higher probability, while the lighter regions contain points with lower probability. Points outside the largest HDR are shown as individual points. Points with lookout probabilities less than 0.05 are optionally shown in red.

Usage

```
gg_hdrboxplot(
  data,
  var1,
  var2 = NULL,
  prob = c(0.5, 0.99),
  color = "#00659e",
  scatterplot = FALSE,
  show_lookout = TRUE,
  ...
)
```

Arguments

`data`	A data frame or matrix containing the data.
`var1`	The name of the first variable to plot (a bare expression).
`var2`	Optionally, the name of the second variable to plot (a bare expression).
`prob`	A numeric vector specifying the coverage probabilities for the HDRs.
`color`	The base color to use for the mode. Colors for the HDRs are generated by whitening this color.
`scatterplot`	A logical argument indicating if a regular HDR plot is required (`FALSE`), or if a scatterplot in the same colors is required (`TRUE`).
`show_lookout`	A logical argument indicating if the plot should highlight observations with "lookout" probabilities less than 0.05.
`...`	Other arguments passed to `kde`.

Details

The original HDR boxplot proposed by Hyndman (1996), R can be produced with all arguments set to their defaults other than `lookout`.

Value

A ggplot object showing an HDR plot or scatterplot of the data.

Author(s)

Rob J Hyndman

References

Hyndman, R J (1996) Computing and Graphing Highest Density Regions, *The American Statistician*, **50**(2), 120–126. https://robjhyndman.com/publications/hdr/ Kandanaarachchi, S & Hyndman, R J (2022) "Leave-one-out kernel density estimates for outlier detection", *J Computational & Graphical Statistics*, **31**(2), 586-599. https://robjhyndman.com/publications/lookout/

Examples

```
df <- data.frame(x = c(rnorm(1000), rnorm(1000, 5, 1)))
df$y <- df$x + rnorm(200, sd=2)
gg_hdrboxplot(df, x)
gg_hdrboxplot(df, x, y, scatterplot = TRUE)
oldfaithful |>
  filter(duration < 7000, waiting < 7000) |>
  gg_hdrboxplot(duration, waiting, scatterplot = TRUE)
cricket_batting |>
  filter(Innings > 20) |>
  gg_hdrboxplot(Average)
```

`glosh_scores` *GLOSH scores*

Description

Compute Global-Local Outlier Score from Hierarchies. This is based on hierarchical clustering where the minimum cluster size is k. The resulting outlier score is a measure of how anomalous each observation is. The function uses `dbscan::hdbscan` to do the calculation.

Usage

```
glosh_scores(y, k = 10, ...)
```

Arguments

y	Numerical matrix or vector of data
k	Minimum cluster size. Default: 5.
...	Additional arguments passed to `dbscan::hdbscan`

Value

Numerical vector containing GLOSH values

Author(s)

Rob J Hyndman

See Also

`dbscan::glosh`

Examples

```
y <- c(rnorm(49), 5)
glosh_scores(y)
```

`grubbs_anomalies` *Statistical tests for anomalies using Grubbs' test and Dixon's test*

Description

Grubbs' test (proposed in 1950) identifies possible anomalies in univariate data using z-scores assuming the data come from a normal distribution. Dixon's test (also from 1950) compares the difference in the largest two values to the range of the data. Critical values for Dixon's test have been computed using simulation with interpolation using a quadratic model on logit(alpha) and log(log(n)).

Usage

```
grubbs_anomalies(y, alpha = 0.05)

dixon_anomalies(y, alpha = 0.05, two_sided = TRUE)
```

Arguments

`y`	numerical vector of observations
`alpha`	size of the test.
`two_sided`	If `TRUE`, both minimum and maximums will be considered. Otherwise only the maximum will be used. (Take negative values to consider only the minimum with `two_sided=FALSE`.)

Details

Grubbs' test is based on z-scores, and a point is identified as an anomaly when the associated absolute z-score is greater than a threshold value. A vector of logical values is returned, where `TRUE` indicates an anomaly. This version of Grubbs' test looks for outliers anywhere in the sample. Grubbs' original test came in several variations which looked for one outlier, or two outliers in one tail, or two outliers on opposite tails. These variations are implemented in the `grubbs.test` function. Dixon's test only considers the maximum (and possibly the minimum) as potential outliers.

Value

A logical vector

Author(s)

Rob J Hyndman

References

Grubbs, F. E. (1950). Sample criteria for testing outlying observations. *Annals of Mathematical Statistics*, 21(1), 27–58. Dixon, W. J. (1950). Analysis of extreme values. *Annals of Mathematical Statistics*, 21(4), 488–506.

See Also

`grubbs.test`, `dixon.test`

Examples

```
x <- c(rnorm(1000), 5:10)
tibble(x = x) |> filter(grubbs_anomalies(x))
tibble(x = x) |> filter(dixon_anomalies(x))
y <- c(rnorm(1000), 5)
tibble(y = y) |> filter(grubbs_anomalies(y))
tibble(y = y) |> filter(dixon_anomalies(y))
```

`hdr_palette` *Color palette designed for plotting Highest Density Regions*

Description

A sequential color palette is returned, with the first color being `color`, and the rest of the colors being a mix of `color` with increasing amounts of white. If `prob` is provided, then the mixing proportions are determined by `prob` (and `n` is ignored). Otherwise the mixing proportions are equally spaced between 0 and 1.

Usage

```
hdr_palette(n, color = "#00659e", prob = NULL)
```

Arguments

`n`	Number of colors in palette.
`color`	First color of vector.
`prob`	Vector of probabilities between 0 and 1.

Value

A function that returns a vector of colors of length `length(prob) + 1`.

Examples

```
hdr_palette(prob = c(0.5, 0.99))
```

`hdr_table` *Table of Highest Density Regions*

Description

Compute the highest density regions (HDR) for a kernel density estimate. The HDRs are returned as a tibble with one row per interval and columns: `prob` (giving the probability coverage), `density` (the value of the density at the boundary of the HDR), For one dimensional density functions, the tibble also has columns `lower` (the lower ends of the intervals), `upper` (the upper ends of the interval), `mode` (the point at which the density is maximized within each interval).

Usage

```
hdr_table(
  y = NULL,
  density = NULL,
  prob = c(0.5, 0.99),
  h = kde_bandwidth(y, method = "double"),
  H = kde_bandwidth(y, method = "double"),
  ...
)
```

Arguments

`y`	Numerical vector or matrix of data
`density`	Probability density function, either estimated by `ks::kde()` or a data frame or matrix with numerical columns that can be passed to `as_kde()`.
`prob`	Probability of the HDR
`h`	Bandwidth for univariate kernel density estimate. Default is `kde_bandwidth`.
`H`	Bandwidth for multivariate kernel density estimate. Default is `kde_bandwidth`.
`...`	If `y` is supplied, other arguments are passed to `kde`. Otherwise, additional arguments are passed to `as_kde`.

Value

A tibble

Author(s)

Rob J Hyndman

References

Hyndman, R J. (1996) Computing and Graphing Highest Density Regions, *The American Statistician*, **50**(2), 120–126.

Examples

```
# Univariate HDRs
y <- c(rnorm(100), rnorm(100, 3, 1))
hdr_table(y = y)
hdr_table(density = ks::kde(y))
x <- seq(-4, 4, by = 0.01)
hdr_table(density = data.frame(y = x, density = dnorm(x)), prob = 0.95)
# Bivariate HDRs
y <- cbind(rnorm(100), rnorm(100))
hdr_table(y = y)
grid <- seq(-4, 4, by=0.1)
density <- expand.grid(grid, grid) |>
  mutate(density = dnorm(Var1) * dnorm(Var2))
hdr_table(density = density)
```

`kde_bandwidth` *Robust bandwidth estimation for kernel density estimation*

Description

Robust bandwidth estimation for kernel density estimation

Usage

```
kde_bandwidth(
  data,
  method = c("robust_normal", "double", "lookout"),
  max.iter = 2
)
```

Arguments

`data`	A numeric matrix or data frame.
`method`	Method to use for selecting the bandwidth. `robust_normal` uses a robust version of the normal reference rule. `lookout` uses the topological data analysis approach that is part of the lookout algorithm.
`max.iter`	How many times should the `lookout` method be iterated. That is, outliers (probability < 0.05) are removed and the bandwidth is re-computed from the remaining observations.

Value

A matrix of bandwidths (or scalar in the case of univariate data).

Author(s)

Rob J Hyndman

Examples

```
# Univariate bandwidth calculation
kde_bandwidth(oldfaithful$duration)
# Bivariate bandwidth calculation
kde_bandwidth(oldfaithful[,2:3])
```

`lof_scores` *Local outlier factors*

Description

Compute local outlier factors using k nearest neighbours. A local outlier factor is a measure of how anomalous each observation is based on the density of neighbouring points. The function uses `dbscan::lof` to do the calculation.

Usage

```
lof_scores(y, k = 10, ...)
```

Arguments

`y`	Numerical matrix or vector of data
`k`	Number of neighbours to include. Default: 5.
`...`	Additional arguments passed to `dbscan::lof`

Value

Numerical vector containing LOF values

Author(s)

Rob J Hyndman

See Also

`dbscan::lof`

Examples

```
y <- c(rnorm(49), 5)
lof_scores(y)
```

`lookout` *Lookout probabilities*

Description

Compute leave-one-out log score probabilities using a Generalized Pareto distribution. These give the probability of each observation being an anomaly.

Usage

```
lookout(
  object = NULL,
  density_scores = NULL,
  loo_scores = density_scores,
  threshold_probability = 0.95
)
```

Arguments

`object`	A model object or a numerical data set.
`density_scores`	Numerical vector of log scores
`loo_scores`	Optional numerical vector of leave-one-out log scores
`threshold_probability`	Probability threshold when computing the POT model for the log scores.

Details

This function can work with several object types. If `object` is not `NULL`, then the object is passed to `density_scores` to compute density scores (and possibly LOO density scores). Otherwise, the density scores are taken from the `density_scores` argument, and the LOO density scores are taken from the `loo_scores` argument. Then the Generalized Pareto distribution is fitted to the scores, to obtain the probability of each observation.

Value

A numerical vector containing the lookout probabilities

Author(s)

Rob J Hyndman

References

Sevvandi Kandanaarachchi & Rob J Hyndman (2022) "Leave-one-out kernel density estimates for outlier detection", *J Computational & Graphical Statistics*, **31**(2), 586-599. `https://robjhyndman.com/publications/lookout/`

Examples

```
# Univariate data
tibble(
  y = c(5, rnorm(49)),
  lookout = lookout(y)
)
# Bivariate data with score calculation done outside the function
tibble(
  x = rnorm(50),
  y = c(5, rnorm(49)),
  fscores = density_scores(y),
  loo_fscores = density_scores(y, loo = TRUE),
  lookout = lookout(density_scores = fscores, loo_scores = loo_fscores)
)
# Using a regression model
of <- oldfaithful |> filter(duration < 7200, waiting < 7200)
fit_of <- lm(waiting ~ duration, data = of)
of |>
  mutate(lookout_prob = lookout(fit_of)) |>
  arrange(lookout_prob)
```

`mvscale` *Compute robust multivariate scaled data*

Description

A multivariate version of `base::scale()`, that takes account of the covariance matrix of the data, and uses robust estimates of center, scale and covariance by default. The centers are removed using medians, the scale function is the IQR, and the covariance matrix is estimated using a robust OGK estimate. The data are scaled using the Cholesky decomposition of the inverse covariance. Then the scaled data are returned. This is useful for computing pairwise Mahalanobis distances.

Usage

```
mvscale(
  object,
  center = stats::median,
  scale = robustbase::s_IQR,
  cov = robustbase::covOGK,
  warning = TRUE
)
```

Arguments

`object`	A vector, matrix, or data frame containing some numerical data.
`center`	A function to compute the center of each numerical variable. Set to NULL if no centering is required.

`scale`	A function to scale each numerical variable. When `cov = robustbase::covOGK`, it is passed as the `sigmamu` argument.
`cov`	A function to compute the covariance matrix. Set to NULL if no rotation required.
`warning`	Should a warning be issued if non-numeric columns are ignored?

Details

Optionally, the centering and scaling can be done for each variable separately, so there is no rotation of the data, by setting `cov = NULL`. Also optionally, non-robust methods can be used by specifying `center = mean`, `scale = stats::sd`, and `cov = stats::cov`. Any non-numeric columns are retained with a warning.

Value

A vector, matrix or data frame of the same size and class as `object`, but with numerical variables replaced by scaled versions.

Author(s)

Rob J Hyndman

Examples

```
# Univariate z-scores (no rotation)
mvscale(oldfaithful, center = mean, scale = sd, cov = NULL, warning = FALSE)
# Non-robust scaling with rotation
mvscale(oldfaithful, center = mean, cov = stats::cov, warning = FALSE)
mvscale(oldfaithful, warning = FALSE)
# Robust Mahalanobis distances
oldfaithful |>
  select(-time) |>
  mvscale() |>
  head(5) |>
  dist()
```

`n01` *Multivariate standard normal data*

Description

A synthetic data set containing 1000 observations on 10 variables generated from independent standard normal distributions.

Usage

```
n01
```

Format

A data frame with 1000 rows and 10 columns.

Value

Data frame

Examples

```
n01
```

`oldfaithful` *Old faithful eruption data*

Description

A data set containing data on recorded eruptions of the Old Faithful Geyser in Yellowstone National Park, Wyoming, USA, from 1 January 2015 to 1 October 2021. Recordings are incomplete, especially during the winter months when observers may not be present.

Usage

```
oldfaithful
```

Format

A data frame with 2261 rows and 3 columns:

time Time eruption started

duration Duration of eruption in seconds

waiting Time to the following eruption

Value

Data frame

Source

https://geysertimes.org

Examples

```
oldfaithful |>
 filter(duration < 7000, waiting < 7000) |>
 ggplot(aes(x = duration, y = waiting)) +
 geom_point()
```

`peirce_anomalies` *Anomalies according to Peirce's and Chauvenet's criteria*

Description

Peirce's criterion and Chauvenet's criterion were both proposed in the 1800s as a way of determining what observations should be rejected in a univariate sample.

Usage

```
peirce_anomalies(y)

chauvenet_anomalies(y)
```

Arguments

`y`	numerical vector of observations

Details

These functions take a univariate sample `y` and return a logical vector indicating which observations should be considered anomalies according to either Peirce's criterion or Chauvenet's criterion.

Value

A logical vector

Author(s)

Rob J Hyndman

References

Peirce, B. (1852). Criterion for the rejection of doubtful observations. *The Astronomical Journal*, 2(21), 161–163.

Chauvenet, W. (1863). 'Method of least squares'. Appendix to *Manual of Spherical and Practical Astronomy*, Vol.2, Lippincott, Philadelphia, pp.469-566.

Examples

```
y <- rnorm(1000)
tibble(y = y) |> filter(peirce_anomalies(y))
tibble(y = y) |> filter(chauvenet_anomalies(y))
```

`stray_anomalies` *Stray anomalies*

Description

Test if observations are anomalies according to the stray algorithm.

Usage

```
stray_anomalies(y, ...)
```

Arguments

`y`	A vector, matrix, or data frame consisting of numerical variables.
`...`	Other arguments are passed to `find_HDoutliers`.

Value

Numerical vector containing logical values indicating if the observation is identified as an anomaly using the stray algorithm.

Author(s)

Rob J Hyndman

Examples

```
# Univariate data
y <- c(6, rnorm(49))
stray_anomalies(y)
# Bivariate data
y <- cbind(rnorm(50), c(5, rnorm(49)))
stray_anomalies(y)
```

`stray_scores` *Stray scores*

Description

Compute stray scores indicating how anomalous each observation is.

Usage

```
stray_scores(y, ...)
```

Arguments

`y`	A vector, matrix, or data frame consisting of numerical variables.
`...`	Other arguments are passed to `find_HDoutliers`.

Value

Numerical vector containing stray scores.

Author(s)

Rob J Hyndman

Examples

```
# Univariate data
y <- c(6, rnorm(49))
scores <- stray_scores(y)
threshold <- stray::find_threshold(scores, alpha = 0.01, outtail = "max", p = 0.5, tn = 50)
which(scores > threshold)
```

`weird_conflicts` *Conflicts between weird packages and other packages*

Description

This function lists all the conflicts between packages in the weird collection and other packages that you have loaded.

Usage

```
weird_conflicts()
```

Details

Some conflicts are deliberately ignored: `intersect`, `union`, `setequal`, and `setdiff` from dplyr; and `intersect`, `union`, `setdiff`, and `as.difftime` from lubridate. These functions make the base equivalents generic, so shouldn't negatively affect any existing code.

Value

A list object of class `weird_conflicts`.

Examples

```
weird_conflicts()
```

`weird_packages` *List all packages loaded by weird*

Description

List all packages loaded by weird

Usage

```
weird_packages(include_self = FALSE)
```

Arguments

`include_self`	Include weird in the list?

Value

A character vector of package names.

Examples

```
weird_packages()
```

Index