Package: weird (via r-universe)

September 17, 2024

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

Version 1.0.2.9000

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, distributional, dplyr (>= 0.7.4), evd, ggplot2 (>= 3.1.1), grDevices, interpolation, ks, lookout, mvtnorm, purrr (>= 0.2.4), rlang, robustbase, rstudioapi (>= 0.7), stray, tibble (>= 1.4.2), vctrs

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

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

RemoteUrl https://github.com/robjhyndman/weird-package

RemoteRef HEAD

RemoteSha 126b6ad0c5360ae1301d7a89722949d0079f81bd

Contents

`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))
```

`dist_density` *Create distributional object based on a specified density*

Description

Creates a distributional object using a density specified as pair of vectors giving (x, f(x)). The density is assumed to be piecewise linear between the points provided, and 0 outside the range of x.

Usage

```
dist_density(x, density)
```

Arguments

`x`	Numerical vector of ordinates, or a list of such vectors.
`density`	Numerical vector of density values, or a list of such vectors.

Examples

```
dist_density(seq(-4, 4, by = 0.01), dnorm(seq(-4, 4, by = 0.01)))
```

`dist_kde` *Create distributional object based on a kernel density estimate*

Description

Creates a distributional object using a kernel density estimate with a Gaussian kernel obtained from the `kde()` function. The bandwidth can be specified; otherwise the `kde_bandwidth()` function is used. The cdf, quantiles and moments are consistent with the kde. Generating random values from the kde is equivalent to a smoothed bootstrap.

Usage

```
dist_kde(y, h = NULL, H = NULL, multiplier = 1, ...)
```

Arguments

`y`	Numerical vector or matrix of data, or a list of such objects. If a list is provided, then all objects should be of the same dimension. e.g., all vectors, or all matrices with the same number of columns.
`h`	Bandwidth for univariate distribution. If `NULL`, the `kde_bandwidth` function is used.
`H`	Bandwidth matrix for multivariate distribution. If `NULL`, the `kde_bandwidth` function is used.

`multiplier`	Multiplier for bandwidth passed to `kde_bandwidth`. Ignored if `h` or `H` are specified.
`...`	Other arguments are passed to `kde`.

Examples

```
dist_kde(c(rnorm(200), rnorm(100, 5)), multiplier = 2)
dist_kde(cbind(rnorm(200), rnorm(200, 5)))
```

`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)
```

`fr_mortality` *French mortality rates by age and sex*

Description

A data set containing French mortality rates between the years 1816 and 1999, by age and sex.

Usage

```
fr_mortality
```

Format

A data frame with 31,648 rows and 4 columns.

Value

Data frame

Source

Human Mortality Database https://www.mortality.org

Examples

```
fr_mortality
```

`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, color = "#00659e", 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).
`color`	The base color to use for the median. Other colors are generated as a mixture of `color` with white.
`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_density` *Produce ggplot of densities from distributional objects in 1 or 2 dimensions*

Description

Produce ggplot of densities from distributional objects in 1 or 2 dimensions

Usage

```
gg_density(
  object,
  prob = seq(9)/10,
  hdr = NULL,
  show_points = FALSE,
  show_mode = FALSE,
  show_anomalies = FALSE,
  colors = c("#0072b2", "#D55E00", "#009E73", "#CC79A7", "#E69F00", "#56B4E9", "#F0E442",
    "#333333"),
  alpha = NULL,
  jitter = FALSE,
  ngrid = 501
)
```

Arguments

`object`	distribution object from the distributional package or `dist_kde`()
`prob`	Probability of the HDRs to be drawn.
`hdr`	Character string describing how the HDRs are to be shown. Options are "none", "fill", "points" and "contours" (the latter only for bivariate plots). If `NULL`, then "none" is used for univariate distributions and "contours" for bivariate.
`show_points`	If `TRUE`, then individual observations are plotted.
`show_mode`	If `TRUE`, then the mode of the distribution is shown as a point.
`show_anomalies`	If `TRUE`, then the observations with surprisal probabilities less than 0.005 are shown in black.
`colors`	Color palette to use. If there are more than `length(colors)` distributions, they are recycled. Default is the Okabe-Ito color palette.
`alpha`	Transparency of points. Ignored if `show_points` is `FALSE`. Defaults to min(1, 500/n), where n is the number of observations plotted.
`jitter`	For univariate distributions, when `jitter` is `TRUE` and `show_points` is TRUE, a small amount of vertical jittering is applied to the observations. Ignored for bivariate distributions.
`ngrid`	Number of grid points to use for the density function.

Details

This function produces a ggplot of a density from a distributional object. 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 ah HDR contour plot of the density function, with the observations optionally shown as points. The mode can also be drawn as a point. The combination of `hdr = "fill"`, `show_points = TRUE`, `show_mode = TRUE`, and `prob = c(0.5, 0.99)` is equivalent to showing HDR boxplots.

Value

A ggplot object.

Author(s)

Rob J Hyndman

Examples

```
# Univariate densities
kde <- dist_kde(c(rnorm(500), rnorm(500, 4, .5)))
gg_density(kde,
  hdr = "fill", prob = c(0.5, 0.95), color = "#c14b14",
  show_mode = TRUE, show_points = TRUE, jitter = TRUE
)
c(dist_normal(), kde) |>
  gg_density(hdr = "fill", prob = c(0.5, 0.95))
# Bivariate density
tibble(y1 = rnorm(5000), y2 = y1 + rnorm(5000)) |>
  dist_kde() |>
  gg_density(show_points = TRUE, alpha = 0.1, hdr = "fill")
```

`gg_density_layer` *Add ggplot layer of densities from distributional objects in 1 dimension*

Description

Add ggplot layer of densities from distributional objects in 1 dimension

Usage

```
gg_density_layer(object, scale = 1, ngrid = 501, ...)
```

Arguments

`object`	distribution object from the distributional package or `dist_kde`()
`scale`	Scaling factor for the density function.
`ngrid`	Number of grid points to use for the density function.
`...`	Additional arguments are passed to `geom_line`.

Details

This function adds a ggplot layer of a density from a distributional object. For univariate densities, it adds a line plot of the density function. For bivariate densities, it adds a contour plot of the density function.

Value

A ggplot layer

Author(s)

Rob J Hyndman

Examples

```
dist_mixture(
  dist_normal(-2, 1),
  dist_normal(2, 1),
  weights = c(1 / 3, 2 / 3)
) |>
  gg_density() +
  gg_density_layer(dist_normal(-2, 1), linetype = "dashed", scale = 1 / 3) +
  gg_density_layer(dist_normal(2, 1), linetype = "dashed", scale = 2 / 3)
```

`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. Observations outside the largest HDR are shown as individual points. Anomalies with leave-one-out surprisal probabilities less than 0.005 are optionally shown in black.

Usage

```
gg_hdrboxplot(
  data,
  var1,
  var2 = NULL,
  prob = c(0.5, 0.99),
  color = "#0072b2",
  show_points = FALSE,
  show_anomalies = TRUE,
  scatterplot = show_points,
  alpha = NULL,
  jitter = TRUE,
  ngrid = 501,
    ...
  )
```

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.
`show_points`	A logical argument indicating if a regular HDR plot is required (`FALSE`), or whether to show the individual observations in the same colors (`TRUE`).
`show_anomalies`	A logical argument indicating if the surprisal anomalies should be shown (in black). These are points with leave-one-out surprisal probability values less than 0.005, and which lie outside the 99% HDR region.
`scatterplot`	Equivalent to `show_points`. Included for compatibility with `gg_bagplot()`.
`alpha`	Transparency of points. Ignored if `show_points` is `FALSE`. Defaults to min(1, 500/n), where n is the number of observations plotted.
`jitter`	A logical value indicating if the points should be vertically jittered for the 1d box plots to reduce overplotting.
`ngrid`	Number of grid points to use for the density function.
`...`	Other arguments passed to `dist_kde`.

Details

The original HDR boxplot proposed by Hyndman (1996), can be produced with `show_anomalies = FALSE`, `jitter = FALSE`, `alpha = 1`, and all other arguments set to their defaults.

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/`

See Also

`surprisal_prob`, `hdr_table`

Examples

```
df <- data.frame(x = c(rnorm(1000), rnorm(1000, 5, 1), 10))
gg_hdrboxplot(df, x, show_anomalies = TRUE)
cricket_batting |>
  filter(Innings > 20) |>
  gg_hdrboxplot(Average)
oldfaithful |>
  filter(duration < 7000, waiting < 7000) |>
  gg_hdrboxplot(duration, waiting, show_points = TRUE)
```

`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))
```

hampel_anomalies *Identify anomalies using the Hampel filter*

Description

The Hampel filter is designed to find anomalies in time series data using mean absolute deviations in the vicinity of each observation.

Usage

```
hampel_anomalies(y, bandwidth, k = 3)
```

Arguments

y	numeric vector containing time series
bandwidth	integer width of the window around each observation
k	numeric number of standard deviations to declare an outlier

Details

First, a moving median is calculated using windows of size 2 * bandwidth + 1. Then the median absolute deviations from this moving median are calculated in the same moving windows. A point is declared an anomaly if its MAD is value is more than k standard deviations. The MAD is converted to a standard deviation using MAD * 1.482602, which holds for normally distributed data. The first bandwidth and last bandwidth observations cannot be declared anomalies.

Value

logical vector identifying which observations are anomalies.

Author(s)

Rob J Hyndman

Examples

```
set.seed(1)
df <- tibble(
  time = seq(41),
  y = c(rnorm(20), 5, rnorm(20))
) |>
  mutate(hampel = hampel_anomalies(y, bandwidth = 3, k = 4))
df |> ggplot(aes(x = time, y = y)) +
  geom_line() +
  geom_point(data = df |> filter(hampel), col = "red")
```

`hdr_table` *Table of Highest Density Regions*

Description

Compute a table of highest density regions (HDR) for a distributional object. 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), and `upper` (the upper ends of the intervals).

Usage

```
hdr_table(object, prob, density_only = FALSE)
```

Arguments

`object`	Distributional object such as that returned by `dist_kde()`
`prob`	Vector of probabilities giving the HDR coverage (between 0 and 1)
`density_only`	If `TRUE`, only the density values are returned for each distribution, not the end points of the interval. (Default `FALSE` for univariate distributions, and `TRUE` for multivariate distributions.)

Value

A tibble

Author(s)

Rob J Hyndman

Examples

```
# Univariate HDRs
c(dist_normal(), dist_kde(c(rnorm(100), rnorm(100, 3, 1)))) |>
  hdr_table(c(0.5, 0.95))
dist_kde(oldfaithful$duration) |> hdr_table(0.95)
# Bivariate HDRs
dist_kde(oldfaithful[, c("duration", "waiting")]) |> hdr_table(0.90)
```

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

Description

Robust bandwidth estimation for kernel density estimation

Usage

```
kde_bandwidth(data, multiplier = 1)
```

Arguments

`data`	A numeric matrix or data frame.
`multiplier`	Bandwidths are chosen using a robust version of the normal reference rule multiplied by a constant. The default is 1.

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_prob *Lookout probabilities*

Description

The lookout algorithm (Kandanaarachchi & Hyndman, 2022) computes leave-one-out surprisal probabilities from a kernel density estimate using a Generalized Pareto distribution. The kernel density estimate uses a bandwidth based on topological data analysis and a quadratic kernel. So it is similar but not identical to using surprisal_prob with `loo = TRUE` and `GPD = TRUE`. A low probability indicates a likely anomaly.

Usage

```
lookout_prob(object, ...)
```

Arguments

`object`	A numerical data set.
`...`	Other arguments are passed to `lookout`.

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/`

See Also

`lookout`

Examples

```
# Univariate data
tibble(
  y = c(5, rnorm(49)),
  lookout = lookout_prob(y)
)
# Bivariate data
tibble(
  x = rnorm(50),
  y = c(5, rnorm(49)),
  lookout = lookout_prob(cbind(x, y))
)
# Using a regression model
of <- oldfaithful |> filter(duration < 7200, waiting < 7200)
fit_of <- lm(waiting ~ duration, data = of)
broom::augment(fit_of) |>
  mutate(lookout = lookout_prob(.std.resid)) |>
  arrange(lookout)
```

`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_Qn,
  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)
```

`surprisals` *Surprisals*

Description

Compute surprisals from a model or a data set given a probability distribution. The surprisals are defined as minus the log of the density at each observation.

Usage

```
surprisals(object, ...)
```

Arguments

`object`	A model or numerical data set
`...`	Other arguments are passed to the appropriate method.

`surprisals.default` *Surprisals computed from a data set*

Description

Compute surprisals from a data set given a probability distribution. The surprisals are defined as minus the log of the density at each observation.

Usage

```
## Default S3 method:
surprisals(
  object,
  distribution = dist_kde(object, multiplier = 2, ...),
  loo = FALSE,
  ...
)
```

Arguments

`object`	A numerical data set, either a vector, matrix or data frame.
`distribution`	A probability distribution stored as a distributional object.
`loo`	Should leave-one-out surprisals be computed?
`...`	Other arguments are passed to `dist_kde`.

Details

If no distribution is provided, a kernel density estimate is computed. The leave-one-out surprisals (or LOO surprisals) are obtained by estimating the kernel density estimate using all other observations.

Value

A numerical vector containing the surprisals.

Author(s)

Rob J Hyndman

See Also

`dist_kde`

Examples

```
# surprisals computed from bivariate data set
oldfaithful |>
  filter(duration < 7000, waiting < 7000) |>
  mutate(
    loo_fscores = surprisals(cbind(duration, waiting), loo = TRUE)
  )
```

`surprisals.lm` *Surprisals computed from a model*

Description

Surprisals computed from a model

Usage

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

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

Arguments

`object`	A model object such as returned by `lm`, or `gam`.
`loo`	Should leave-one-out surprisals be computed?
`...`	Other arguments are ignored.

Examples

```
# surprisals computed from linear model
of <- oldfaithful |>
  filter(duration < 7200, waiting < 7200)
lm_of <- lm(waiting ~ duration, data = of)
of |>
  mutate(
    fscore = surprisals(lm_of),
    loo_fscore = surprisals(lm_of, loo = TRUE),
    # lookout_prob = lookout(surprisals = fscore, loo_scores = loo_fscore)
  ) |>
  ggplot(aes(
    x = duration, y = waiting,
    color = loo_fscore > quantile(loo_fscore, 0.99)
  )) +
  geom_point()
# surprisals computed from GAM
of <- oldfaithful |>
  filter(duration > 1, duration < 7200, waiting < 7200)
gam_of <- mgcv::gam(waiting ~ s(duration), data = of)
of |>
  mutate(fscore = surprisals(gam_of))
```

`surprisal_prob` *Surprisal probabilities*

Description

Compute the probability of a surprisal at least as extreme as those observed. A surprisal is given by $-\log(f)$ where f is the density or probability mass function of the distribution. The surprisal values are computed from the `distribution` provided. If no distribution is provided, a kernel density estimate is used.

Usage

```
surprisal_prob(
  object,
  distribution = NULL,
  loo = FALSE,
  GPD = FALSE,
  threshold_probability = 0.1,
  ...
)
```

Arguments

`object`	A model or numerical data set.
`distribution`	A probability distribution stored as a distributional object. Ignored if `object` is a model.
`loo`	Logical value specifying if leave-one-out surprisals should be computed.
`GPD`	Logical value specifying if a Generalized Pareto distribution should be used to estimate the probabilities.
`threshold_probability`	Probability threshold when computing the GPD distribution for the surprisals.
`...`	Other arguments are passed to `surprisals`.

Details

The surprisal probabilities may be computed in three different ways.

1. Given the same `distribution` that was used to compute the surprisal values. Under this option, surprisal probabilities are equal to 1 minus the coverage probability of the largest HDR that contains each value. Surprisal probabilities smaller than 1e-6 are returned as 1e-6.
2. Using a Generalized Pareto Distribution fitted to the most extreme surprisal values (those with probability less than `threshold_probability`). This option is used if `GPD = TRUE`. For surprisal values with probability less than `threshold_probability`, the value of `threshold_probability` is returned. Under this option, the distribution is used for computing the surprisal values but not for determining their probabilities. Due to extreme value theory, the resulting probabilities should be relatively insensitive to the distribution used in computing the surprisal values.
3. Empirically as the proportion of observations with greater surprisal values. This option is used when `GPD = FALSE` and no distribution is explicitly provided. This is also insensitive to the distribution used in computing the surprisal values.

Examples

```
# Univariate data
tibble(
  y = c(5, rnorm(49)),
  p = surprisal_prob(y)
)
tibble(
  y = n01$v1,
  prob1 = surprisal_prob(y),
  prob2 = surprisal_prob(y, GPD = TRUE),
  prob3 = surprisal_prob(y, dist_normal()),
  prob4 = surprisal_prob(y, dist_normal(), GPD = TRUE)
) |>
  arrange(prob1)
# Bivariate data
tibble(
  x = rnorm(50),
  y = c(5, rnorm(49)),
  lookout = lookout_prob(cbind(x, y))
)
# Using a regression model
of <- oldfaithful |> filter(duration < 7200, waiting < 7200)
fit_of <- lm(waiting ~ duration, data = of)
of |>
  mutate(p = surprisal_prob(fit_of)) |>
  arrange(p)
```

`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