Package: weird (via r-universe)

November 20, 2026

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

Description All functions and data sets required for the examples in the book Hyndman (2026) ``That's Weird: Anomaly Detection Using R'' <https://OTexts.com/weird/>. All packages needed to run the examples are also loaded.

Version 3.2.0

Depends R (>= 4.1.0)

Imports aplpack, broom, cli (>= 3.6.1), dbscan, distributional, dplyr (>= 0.7.4), evd, ggplot2 (>= 3.1.1), grDevices, ks, mlpack, RANN, robustbase, stray, vctrs

Suggests knitr, mgcv, mclust, rmarkdown, rrcov, testthat (>= 3.0.0), tidyr

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

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

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Config/Needs/website tidyverse/tidytemplate

NeedsCompilation no

Author Rob Hyndman [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-2140-5352>), Torben Tvedebrink [ctb], Posit Software, PBC [cph]

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

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

Date/Publication 2026-11-20 14:30:02 UTC

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

RemoteRef HEAD

RemoteSha 6e71e628859e87694059ce1bc2445c40c0f8a125

Contents

augment.Pca . . . 3
biplot_projection . . . 4
cricket_batting . . . 5
density_df . . . 6
dist_kde . . . 7
dist_mclust . . . 8
fashion . . . 8
fetch_air_quality . . . 9
fetch_oz_books . . . 11
fetch_pets2006 . . . 12
fetch_wine_reviews . . . 13
fr_mortality . . . 14
gg_bagplot . . . 15
gg_density . . . 16
gg_hdrboxplot . . . 17
glosh_scores . . . 19
grubbs_anomalies . . . 20
gun_deaths . . . 21
hampel_anomalies . . . 22
hdr_regions . . . 23
hdr_table . . . 24
kde_bandwidth . . . 25
lof_scores . . . 26
mvscale . . . 27
n01 . . . 29
oldfaithful . . . 29
outlier_map . . . 30
peirce_anomalies . . . 32
road_deaths . . . 33
stray_anomalies . . . 34
stray_scores . . . 35
surprisals . . . 36
surprisals.factor . . . 37
surprisals.lm . . . 39
surprisals.numeric . . . 42
us_mortality . . . 44
us_senate . . . 46

Index **48**

`augment.Pca` *Augment data with results from a robust principal component analysis*

Description

Augment the data with information from an `rrcov::Pca*` object (such as the output of `rrcov::PcaHubert()` or `rrcov::PcaClassic()`). The returned tibble contains the principal component scores (`.fittedPC1`, `.fittedPC2`, ...), the score distance (`.sd`) and the orthogonal distance (`.od`) of each observation. The score distance measures how far an observation lies from the centre *within* the projection subspace, while the orthogonal distance measures how far it lies *from* the subspace. If `data` is supplied, its columns are returned alongside these results.

Usage

```
## S3 method for class 'Pca'
augment(x, data = NULL, ...)
```

Arguments

`x`	An `rrcov::Pca*` object.
`data`	The original data matrix or data frame used to compute the PCA. If supplied, its columns are bound to the left of the returned tibble.
`...`	Unused.

Value

A `tibble::tibble()` with one row per observation.

Author(s)

Rob J Hyndman

Examples

```
Y <- oldfaithful[, c("duration", "waiting")]
pca <- rrcov::PcaHubert(as.matrix(Y), k = 1)
broom::augment(pca, data = Y)
```

`biplot_projection` *Biplot of a two-dimensional projection*

Description

Draw a two-dimensional projection of the scores with the original variable axes overlaid as arrows (loadings), as in a biplot. Pass `object` (the output of `stats::prcomp()` or an `rrcov::Pca*` function); otherwise supply `scores` and `loadings` directly. All scores should be centred about the origin. The arrows are stretched by a common factor so that the longest arrow just reaches the edge of the point cloud, and only loadings longer than `label_threshold` are labelled.

Usage

```
biplot_projection(
  object = NULL,
  scores = NULL,
  loadings = NULL,
  label_threshold = 0,
  arrow_colour = "#c14b14",
  ...
)
```

Arguments

`object`	Optionally, the output of `stats::prcomp()` or an `rrcov::Pca*` function. If supplied, the scores and loadings are extracted from it and the `scores` and `loadings` arguments are ignored.
`scores`	A matrix or data frame of scores centred about the origin, with the first two columns used as the horizontal and vertical coordinates. Ignored if `object` is supplied.
`loadings`	A matrix or data frame of loadings, with row names giving the variable names and the first two columns used as the arrow directions. Ignored if `object` is supplied.
`label_threshold`	Only loadings whose absolute length exceeds this threshold are labelled. The default of `0` labels every non-zero loading.
`arrow_colour`	Colour of the arrows and labels.
`...`	Additional arguments passed to `ggplot2::geom_point()`.

Value

A `ggplot` object.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 9, https://OTexts.com/weird/highdim.html.

Examples

```
oldfaithful[, c("duration", "waiting")] |>
  prcomp(scale = TRUE) |>
  biplot_projection()
```

---

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 2025.

Usage

```
cricket_batting
```

Format

A data frame with 3968 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/

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 1.4, https://OTexts.com/weird/.

Examples

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

---

density_df *Convert distributional object to a data frame*

Description

Make a long-form data frame containing densities from a distributional object on a regular grid for plotting.

Usage

```
density_df(object, ngrid = NULL)
```

Arguments

object	A distributional object
ngrid	Number of grid points in each dimension. Defaults to 501 for univariate distributions and 101 for bivariate distributions.

Value

Data frame with columns x, y (if bivariate), density, and distribution.

Examples

```
dist_kde(oldfaithful$duration) |> density_df()
```

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,
  method = c("robust", "normal", "plugin", "scv", "lookout"),
  ...
)
```

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. Ignored if y has 2 or more columns. If NULL, the kde_bandwidth function is used.
H	Bandwidth matrix for multivariate distribution. If NULL, the kde_bandwidth function is used.
method	The method of bandwidth estimation to use. See kde_bandwidth() for details. Ignored if h or H are specified.
...	Other arguments are passed to kde.

Value

A distributional object of class dist_kde.

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.9 and 3.9, https://OTexts.com/weird/.

Examples

```
dist_kde(c(rnorm(200), rnorm(100, 5)))
dist_kde(cbind(rnorm(200), rnorm(200, 5)), binned = TRUE)
```

dist_mclust *Convert Gaussian mixture model to a distributional object*

Description

Convert Gaussian mixture model to a distributional object

Usage

```
dist_mclust(object)
```

Arguments

object	Object of class Mclust, output from the [mclust::Mclust] function

Value

An object of class distributional

Examples

```
library(mclust, exclude = "count")
# Univariate mixture density
gmm <- Mclust(oldfaithful$duration) |> dist_mclust()
gg_density(gmm) +
  geom_jitter(data = oldfaithful, aes(x=duration, y = -0.0002),
    width=0, height=0.0002, alpha = 0.1) +
  labs(x = "Eruption duration", y = "")
```

fashion *Fashion-MNIST sneakers*

Description

All 1000 images of sneakers from the Fashion-MNIST test set, together with 10 images from other classes of varying similarity to sneakers: four ankle boots, three sandals, two bags and one pair of trousers. Each image is 28 x 28 pixels.

Usage

```
fashion
```

Format

A data frame with 1010 rows and 4 columns:

id Index of the image in the Fashion-MNIST test set

label Class of the image (a factor with the 10 Fashion-MNIST classes)

planted `TRUE` for the 10 images that are not sneakers

pixels A 1010 x 784 integer matrix of pixel intensities, from 0 (white) to 255 (black), with one row per image. Each row contains the 28 x 28 pixels of an image, stored row by row, so `matrix(pixels[i, ], nrow = 28, byrow = TRUE)` gives image `i`.

Value

Data frame

Source

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. https://github.com/zalandoresearch/fashion-mnist (MIT licence)

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 14, https://OTexts.com/weird/imagevideo.html.

Examples

```
fashion |>
  count(label)
```

---

fetch_air_quality *Air quality data for 12 Beijing monitoring stations from 2013 to 2017*

Description

Hourly air quality measurements from 12 monitoring stations across Beijing, China, from 1 March 2013 to 28 February 2017. The data are downloaded and returned.

Usage

```
fetch_air_quality()
```

Format

A data frame with 420,768 rows and 17 columns:

station Name of the monitoring station

year Year of measurement

month Month of measurement

day Day of measurement

hour Hour of measurement (0–23)

pm2_5 Particulate matter with diameter less than 2.5 micrometers (micrograms per cubic meter)

pm10 Particulate matter with diameter less than 10 micrometers (micrograms per cubic meter)

so2 Sulfur dioxide concentration (micrograms per cubic meter)

no2 Nitrogen dioxide concentration (micrograms per cubic meter)

co Carbon monoxide concentration (micrograms per cubic meter)

o3 Ozone concentration (micrograms per cubic meter)

temperature Temperature (degrees Celsius)

pressure Atmospheric pressure (hPa)

dew_point Dew point temperature (degrees Celsius)

rainfall Rainfall (millimeters)

wind_direction Wind direction

wind_speed Wind speed (meters per second)

Value

Data frame

Source

Chen, S. (2017). Beijing Multi-Site Air Quality Dataset. UCI Machine Learning Repository. doi:10.24432/C5RK5G

References

Hyndman, R J (2026) *That's weird: Anomaly detection using R*, https://OTexts.com/weird/.

Examples

```
## Not run:
air_quality <- fetch_air_quality()
air_quality |>
  filter(station == "Aotizhongxin") |>
  ggplot(aes(x = temperature, y = pm2_5)) +
  geom_point(alpha = 0.1)

## End(Not run)
```

`fetch_oz_books` *Oz books by L. Frank Baum and Ruth Plumly Thompson*

Description

The text of 29 Oz books: the 14 written by L. Frank Baum (1900–1920), and 15 of those written by Ruth Plumly Thompson (1921–1939). Each row is a paragraph. Front matter, chapter headings and titles, illustration captions and back matter have been removed, and curly quotes replaced by straight quotes. The data are downloaded and returned.

Usage

```
fetch_oz_books()
```

Format

A data frame with 31,258 rows and 7 columns:

title Title of the book

author Author of the book

year Year of first publication

gutenberg_id Project Gutenberg ebook number

chapter Chapter number

paragraph Paragraph number within the book

text Text of the paragraph

Details

The Royal Book of Oz (1921) was credited to Baum, who had died in 1919, but was written by Thompson. *The Magic of Oz* (1919) and *Glinda of Oz* (1920) were published after Baum's death. Thompson's Oz books from 1931 to 1934 are not included as they are not available from Project Gutenberg.

Value

Data frame

Source

Project Gutenberg, https://www.gutenberg.org. Downloaded 9 October 2026.

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 11, https://OTexts.com/weird/text.html.

Examples

```
## Not run:
oz_books <- fetch_oz_books()
oz_books |>
  group_by(title, author, year) |>
  summarise(paragraphs = n(), .groups = "drop") |>
  arrange(year)

## End(Not run)
```

`fetch_pets2006` *PETS2006 video frames*

Description

Greyscale frames from the PETS2006 video in the CDnet 2014 change detection benchmark. The video was taken by a fixed camera overlooking the concourse of a railway station. For most of the video, a man stands near the glass wall at the back of the concourse with a bag, while other people walk past. He then puts the bag down, and eventually walks away, leaving it behind. The original 720 x 576 colour frames have been converted to greyscale and reduced to 120 x 96 pixels by averaging over blocks of 6 x 6 pixels. The data are downloaded and returned.

Usage

```
fetch_pets2006()
```

Format

A data frame with 1200 rows and 2 columns:

frame Frame number

pixels A 1200 x 11520 integer matrix of pixel intensities, from 0 (black) to 255 (white), with one row per frame. Each row contains the 96 x 120 pixels of a frame, stored column by column, so `matrix(pixels[i, ], nrow = 96)` gives frame `i` as an image.

Details

The benchmark uses frames 1–299 to initialise background models, and evaluates methods on frames 300–1200.

Value

Data frame

Source

Wang, Y., Jodoin, P.-M., Porikli, F., Konrad, J., Benezeth, Y., & Ishwar, P. (2014). CDnet 2014: An expanded change detection benchmark dataset. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 393–400. http://changedetection.net

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 14, https://OTexts.com/weird/imagevideo.html.

Examples

```
## Not run:
pets2006 <- fetch_pets2006()
# Show frame 951
image(t(matrix(pets2006$pixels[951, ], nrow = 96))[, 96:1], col = grey.colors(256))

## End(Not run)
```

---

fetch_wine_reviews *Wine prices and points*

Description

A data set containing data on wines from 42 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 111,592 rows and 9 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

year Year of wine

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

price Price of a bottle of wine in $US

description Text of the review

Value

Data frame

Source

https://www.kaggle.com/datasets/zynicide/wine-reviews

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 1.4, https://OTexts.com/weird/.

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 31648 rows and 4 columns.

Value

Data frame

Source

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

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 12, https://OTexts.com/weird/timeseries.html.

Examples

```
fr_mortality
```

gg_bagplot *Bagplot*

Description

Produces a bivariate bagplot. A bagplot is analogous 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", show_points = 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.
show_points	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.

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 5.6, https://OTexts.com/weird/.

See Also

bagplot

Examples

```
gg_bagplot(n01, v1, v2)
gg_bagplot(n01, v1, v2, show_points = 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 = NULL
)
```

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 (using a GPD approximation) 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 in each dimension, passed to density_df(). Defaults to 501 for univariate distributions and 101 for bivariate distributions.

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 an 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, 0.5)))
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_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,
  alpha = NULL,
  jitter = 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.
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 (using a GPD approximation), and which lie outside the 99% HDR region.
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.
...	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/

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 5.7, https://OTexts.com/weird/.

See Also

surprisals, 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 |>
  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, using core distances to the k-th nearest neighbour. 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	Number of neighbours to include, not counting the observation itself. Default: 10.
...	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.

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 4.4-4.5, https://OTexts.com/weird/.

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))
```

---

gun_deaths *Gun ownership and homicide rates by country*

Description

A data set containing gun ownership rates and homicide rates for 2017 for various countries around the world. The gun ownership rates are the number of guns owned by civilians per 100 people. The homicide rates are the number of homicides per 100,000 people where the weapon was a firearm.

Usage

```
gun_deaths
```

Format

A data frame with 77 rows and 4 columns:

country Country name

region World region according to Our World in Data

gun_ownership_rate Gun ownership rate (number of guns owned by civilians per 100 people)

homicide_rate Homicide rate (number of homicides per 100,000 people where the weapon was a firearm)

Value

Data frame

Source

World Population Review https://worldpopulationreview.com/country-rankings/gun-ownership-by-country and https://ourworldindata.org/grapher/homicide-rates-from-firearms

Examples

```
gun_deaths
```

`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,
  alpha = NULL,
  approximation = c("none", "gpd", "empirical")
)
```

Arguments

`y`	numeric vector containing time series
`bandwidth`	integer specifying half-width of the window around each observation. That is, each window contains observations from t-bandwidth, ..., t+bandwidth. Must be at least 1.
`k`	numeric number of standard deviations to declare an anomaly. Ignored if `alpha` is specified.
`alpha`	numeric significance level for declaring an anomaly under a normal distribution. If specified, `k` is ignored and the threshold is determined by the significance level.
`approximation`	character string specifying the method to use for approximating the tail of the distribution of surprisal values. Options are "none" (no approximation), "gpd" (generalized Pareto distribution), or "empirical" (empirical distribution).

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 value is more than `k` standard deviations. The MAD is converted to a standard deviation using MAD * 1.4826, 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

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 12.2, https://OTexts.com/weird/timeseries.html.

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_regions *Highest density regions for each observation*

Description

For a `dist_kde` object, determine which highest density region (HDR) each observation falls in, for one or more coverage probabilities. Some densities are multimodal, so the HDR for a given `prob` can consist of several disjoint regions; the returned columns identify the specific region (`1`, `2`, ...) that each observation falls in, ordered from lowest to highest along the first variable, with `NA` for observations outside the HDR at that `prob`.

Usage

```
hdr_regions(object, prob)
```

Arguments

`object`	A `dist_kde` object, as returned by `dist_kde()`, containing a single distribution estimated from univariate or bivariate data.
`prob`	A numeric vector of probabilities giving the HDR coverage (between 0 and 1).

Value

A tibble containing the data used to estimate `object`, along with one additional integer column per element of `prob` (named `hdr_<100*prob>`) showing which region of the corresponding HDR each observation falls in.

Author(s)

Rob J Hyndman

See Also

`hdr_table`, `gg_hdrboxplot`

Examples

```
dist_kde(oldfaithful$duration) |> hdr_regions(c(0.5, 0.95))
dist_kde(oldfaithful[, c("duration", "waiting")]) |> hdr_regions(0.90)
```

---

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)
```

Arguments

`object`	Distributional object such as that returned by `dist_kde()`
`prob`	Vector of probabilities giving the HDR coverage (between 0 and 1)

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. https://robjhyndman.com/publications/hdr/

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.7, 3.4. https://OTexts.com/weird/.

See Also

gg_hdrboxplot

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

Bandwidth matrices are estimated in several ways including a normal reference rule, a robust version of the normal reference rule (default), a plugin estimator, a smoothed cross-validation estimator, or using the approach of Hyndman, Kandanaarachchi & Turner (2026). Details of each method are given in Hyndman (2026).

Usage

```
kde_bandwidth(
  data,
  method = c("robust", "normal", "plugin", "scv", "lookout"),
  ...
)
```

Arguments

data	A numeric matrix or data frame.
method	A character string giving the method to use. Possibilities are: "normal" (normal reference rule), "robust" (a robust version of the normal reference rule, the default), "plugin" (a plugin estimator), "scv" (a smoothed cross-validation estimator), and "lookout" (the bandwidth matrix estimate of Hyndman, Kandanaarachchi & Turner, 2026).
...	Additional arguments are ignored.

Value

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

Author(s)

Rob J Hyndman

References

Hyndman, R J, Kandanaarachchi, S & Turner, K (2026) "When lookout sees crackle: Anomaly detection via kernel density estimation", unpublished. https://robjhyndman.com/publications/lookout2.html

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.9 and 3.9, https://OTexts.com/weird/.

Examples

```
# Univariate bandwidth calculation
kde_bandwidth(oldfaithful$duration)
# Bivariate bandwidth calculation
kde_bandwidth(oldfaithful[, c("duration", "waiting")])
```

---

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, not counting the observation itself. Default: 10.
...	Additional arguments passed to `dbscan::lof`

Value

Numerical vector containing LOF values. An observation has an infinite LOF when its neighbourhood includes at least `k + 1` identical observations (whose local reachability density is infinite) but it is not one of them; the identical observations themselves have LOF values of 1.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 6.6, https://OTexts.com/weird/.

See Also

dbscan::lof

Examples

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

`mvscale` *Compute robust multivariate scaled data*

Description

A multivariate version of base::scale(), that takes account of the covariance matrix of the data. By default, robust estimates are used: the centers are removed using medians, the scale function for univariate data is s_Qn, and the covariance matrix for multivariate data is estimated using a robust MCD estimate. The data are scaled using the Cholesky decomposition of the inverse (co)variance. Then the scaled data are returned. Details of the methods are provided by Hyndman (2026).

Usage

```
mvscale(
  object,
  center = stats::median,
  scale = robustbase::s_Qn,
  cov = robustbase::covMcd,
  alpha = 0.9,
  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(), scale is passed as the sigmamu argument. When cov = robustbase::covMcd(), scale is passed as the scalefn argument.
cov	A function to compute the covariance matrix. Set to NULL if no rotation required. cov() must either return the matrix directly, or a list containing a matrix named cov.
alpha	When cov = robustbase::covMcd(), alpha controls the size of the subsets over which the determinant is minimized. Otherwise it is ignored. Set to 0.9 by default.
warning	Should a warning be issued if non-numeric columns are ignored?
...	Other arguments are passed to cov().

Details

Optionally, the centering and scaling can be done for each variable separately, by setting cov = NULL, so there is no rotation of the data, 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. Missing values are removed before the centers, scale and cov are estimated.

Value

A vector, matrix or data frame of the same size and class as object, but with numerical variables replaced by scaled versions (renamed if they have been rotated).

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.8, 3.6 and 3.7.

See Also

base::scale(), stats::sd(), stats::cov(), robustbase::covMcd(), robustbase::covOGK(), robustbase::s_Qn()

Examples

```
# Univariate z-scores
z <- mvscale(oldfaithful$duration, center = mean, scale = sd)
# Non-robust scaling with no rotation
oldfaithful |>
  mvscale(center = mean, scale = sd, cov = NULL, warning = FALSE)
# Non-robust scaling with rotation
oldfaithful |>
  mvscale(center = mean, scale = sd, cov = stats::cov, warning = FALSE)
# Robust scaling and rotation
oldfaithful |>
  mvscale(warning = FALSE)
```

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

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 1.4, https://OTexts.com/weird/.

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 14 January 2017 to 29 December 2023. Recordings are incomplete, especially during the winter months when observers may not be present.

Usage

```
oldfaithful
```

Format

A data frame with 2097 rows and 4 columns:

time Time eruption started

recorded_duration Duration of eruption as recorded

duration Duration of eruption in seconds

waiting Time to the following eruption in seconds

Value

Data frame

Source

https://geysertimes.org

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 1.4, https://OTexts.com/weird/.

Examples

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

---

outlier_map *Outlier map from a projection or principal component analysis*

Description

Draw an outlier map showing the score distance and orthogonal distance of each observation from a projection or principal component analysis. The score distance measures how far an observation lies from the centre *within* the projection subspace, while the orthogonal distance measures how far it lies *from* the subspace. Pass `object` (the output of `stats::prcomp()` or an rrcov `Pca*` function); otherwise supply `scores` and `loadings` together with the original `data`. For a `prcomp` object, use its `rank.` argument to set the number of retained components.

When `object` is a PCA-like object and `show_thresholds = TRUE`, the score-distance and orthogonal-distance cutoffs are drawn as dashed lines and observations are coloured by type:

Regular observation small score and orthogonal distance.

Good leverage point large score distance, small orthogonal distance.

Orthogonal outlier small score distance, large orthogonal distance.

Bad leverage point large score and orthogonal distance.

The cutoffs are only defined for PCA-like objects, so `show_thresholds` is ignored when `scores` and `loadings` are passed directly.

Usage

```
outlier_map(
  object = NULL,
  data = NULL,
  scores = NULL,
  loadings = NULL,
  show_thresholds = TRUE,
  ...
)
```

Arguments

object	Optionally, the output of stats::prcomp() or an rrcov Pca* function. For a prcomp object, set the number of retained components with its rank. argument. If supplied, the scores and loadings arguments are ignored.
data	The original data matrix or data frame used to compute the projection, scaled if the projection was computed on scaled data. This is required to compute the orthogonal distances, except when object is a rrcov Pca* object (which stores them).
scores	A matrix or data frame of scores, with one column per retained component. Ignored if object is supplied.
loadings	A matrix or data frame of loadings, with one column per retained component. Ignored if object is supplied.
show_thresholds	If TRUE (the default) and object is a PCA-like object, the score-distance and orthogonal-distance cutoffs are drawn as dashed lines and observations are coloured by type. Ignored when scores and loadings are passed directly.
...	Additional arguments passed to ggplot2::geom_point().

Value

A ggplot object.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 9, https://OTexts.com/weird/highdim.html.

Examples

```
Y <- oldfaithful[, c("duration", "waiting")]
prcomp(Y, scale = TRUE, rank. = 1) |>
  outlier_map(data = Y)
```

`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.

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 4.3, https://OTexts.com/weird/.

Examples

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

road_deaths *Australian road deaths*

Description

Every death from a road crash in Australia from January 1989 to December 2025, from the Australian Road Deaths Database (ARDD). Each row is a person killed; people killed in the same crash share a `crash_id`. A death is counted if the crash occurred on a public road, was unintentional, and the person died within 30 days.

Usage

```
road_deaths
```

Format

A data frame with 58168 rows and 20 columns:

crash_id Crash identifier

state State or territory where the crash occurred

year Year of crash

month Month of crash (1–12)

day_of_week Day of the week of the crash

time Time of crash in hours after midnight (e.g. 14.5 is 2:30pm)

crash_type Single or multiple vehicle crash

bus Was a bus involved?

heavy_rigid_truck Was a heavy rigid truck involved?

articulated_truck Was an articulated truck involved?

speed_limit Posted speed limit at the crash location (km/h)

road_user Road user type of the person killed

gender Gender of the person killed

age Age of the person killed (years)

remoteness ABS remoteness area of the crash location (ASGS 2021)

sa4 ABS Statistical Area Level 4 of the crash location (ASGS 2021)

lga Local government area of the crash location (ASGS 2021)

road_type Type of road

christmas Did the crash occur in the 12 days from 23 December?

easter Did the crash occur in the 5 days from the Thursday before Good Friday?

Details

Missing values (coded as -9 or "Unknown" in the source) are NA. Some variables were not recorded for early years: heavy_rigid_truck is mostly missing before 2002, and remoteness, sa4, lga and road_type are missing before 2014 and incomplete until 2017.

Value

Data frame

Source

Bureau of Infrastructure and Transport Research Economics (2026). Australian Road Deaths Database, fatalities. Downloaded 9 October 2026. Licensed under CC BY. https://catalogue.data.infrastructure.gov.au/dataset/australian-road-deaths-database

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 10, https://OTexts.com/weird/categorical.html.

Examples

```
road_deaths |>
  ggplot(aes(x = age, fill = road_user)) +
  geom_histogram(binwidth = 1)
```

---

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

References

Talagala, P D, Hyndman, R J, and Smith-Miles, K (2021) Anomaly detection in high-dimensional data, *Journal of Computational and Graphical Statistics*, **30**(2), 360-374.

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

References

Talagala, P D, Hyndman, R J, and Smith-Miles, K (2021) Anomaly detection in high-dimensional data, *Journal of Computational and Graphical Statistics*, **30**(2), 360-374.

Examples

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

`surprisals` *Surprisals and surprisal probabilities*

Description

A surprisal is given by $s = -\log f(y)$ where f is the density or probability mass function of the estimated or assumed distribution, and y is an observation. This is returned by `surprisals()`. A surprisal probability is the probability of a surprisal at least as extreme as s. This is returned by `surprisals_prob()`

Usage

```
surprisals(object, ...)

surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  ...
)
```

Arguments

`object`	A model or numerical data set
`...`	Other arguments are passed to the appropriate method.
`approximation`	Character string specifying the method to use in computing the surprisal probabilities. See Details below.
`threshold_probability`	Probability threshold when computing the GPD approximation. This is the probability below which the GPD is fitted. Only used if `approximation = "gpd"`.

Details

The surprisal probabilities may be computed in three different ways.

1. When `approximation = "none"` (the default), the surprisal probabilities are computed using 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. When `approximation = "gdp"`, the surprisal probabilities are computed using a Generalized Pareto Distribution fitted to the most extreme surprisal values (those with probability less than `threshold_probability`). For surprisal probabilities greater 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. When `approximation = "empirical"` (or `"rank"`), the surprisal probability of each observation is estimated using the proportion of observations with greater or equal surprisal values. This is a nonparametric approach that is also insensitive to the distribution used in computing the surprisal values.

Value

A numerical vector containing the surprisals or surprisal probabilities.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 7, https://OTexts.com/weird/density.html.

Hyndman, R J & Frazier, D T (2026) "Anomaly detection using surprisals", https://robjhyndman.com/publications/surprisals.html.

See Also

For specific methods, see `surprisals.numeric()` and `surprisals.lm()`,

`surprisals.factor` *Surprisals and surprisal probabilities of categorical data*

Description

For a categorical variable taking one of the categories $c_1, \dots, c_K$, the surprisal of an observation y is $s = -\log f(y)$, where $f(c_k)$ is the probability of category c_k. The surprisal probability is the total probability of all categories that are no more probable than y:

$$p = \sum_{k=1}^{K} f(c_k) 1\big(f(c_k) \leq f(y)\big).$$

Tied categories (with the same probability) have the same surprisal probability, which includes the probabilities of all of them.

Usage

```
## S3 method for class 'factor'
surprisals(object, prob = NULL, prior = 0, loo = FALSE, ...)

## S3 method for class 'factor'
surprisals_prob(
  object,
  approximation = c("none", "empirical", "rank"),
  threshold_probability = 0.1,
  prob = NULL,
  prior = 0,
  loo = FALSE,
  ...
)
```

Arguments

`object`	A factor.
`prob`	A numeric vector of probabilities for the levels of `object`, either named by level or in the order of `levels(object)`. If `NULL` (the default), the probabilities are estimated from `object`.
`prior`	A non-negative constant a added to the count of each level when estimating the probabilities. Only used when `prob` is `NULL`.
`loo`	Should leave-one-out surprisals be computed? Only used when `prob` is `NULL`.
`...`	Other arguments are ignored.
`approximation`	Character string specifying the method to use in computing the surprisal probabilities. With `"none"` (the default), the exact probabilities are computed from the category probabilities. With `"empirical"` (or `"rank"`), the surprisal probability of each observation is the proportion of observations with greater or equal surprisals.
`threshold_probability`	Not used for categorical data.

Details

By default, the probabilities are estimated from the data as $\hat{f}(c_k) = (n_k + a)/(n + Ka)$, where n_k is the number of observations in category c_k, n is the number of non-missing observations, K is the number of levels of `object` (including levels with no observations), and a is `prior`. This is the posterior mean of $f(c_k)$ under a symmetric Dirichlet prior with parameter a. With `loo = TRUE`, each observation is omitted when estimating the probabilities used for its own surprisal and surprisal probability, so that n_k and n are each reduced by one for an observation in category c_k.

To compute surprisals of several categorical variables jointly, combine them into a single factor with `interaction()`.

Value

A numerical vector containing the surprisals or surprisal probabilities.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 7, https://OTexts.com/weird/density.html.

Hyndman, R J & Frazier, D T (2026) "Anomaly detection using surprisals", https://robjhyndman.com/publications/surprisals.html.

Examples

```
# Road types of Australian road deaths since 2014
road_deaths |>
  filter(!is.na(road_type)) |>
  mutate(
    s = surprisals(road_type),
    p = surprisals_prob(road_type)
  ) |>
  count(road_type, s, p) |>
  arrange(p)
# Joint surprisal probabilities of road user and sex
road_deaths |>
  filter(!is.na(road_user), !is.na(gender)) |>
  mutate(p = surprisals_prob(interaction(road_user, gender))) |>
  count(road_user, gender, p) |>
  arrange(p)
```

---

surprisals.lm *Surprisals and surprisal probabilities computed from a model*

Description

A surprisal is given by $s = -\log f(y)$ where f is the density or probability mass function of the estimated or assumed distribution, and y is an observation. This is returned by `surprisals()`. A surprisal probability is the probability of a surprisal at least as extreme as s. This is returned by `surprisals_prob()`

Usage

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

## S3 method for class 'glm'
surprisals(object, ...)

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

```
## S3 method for class 'lm'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  loo = FALSE,
  ...
)

## S3 method for class 'glm'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  ...
)

## S3 method for class 'gam'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  ...
)
```

Arguments

`object`	A model object such as returned by `lm`, `glm`, or `gam`. This includes a specified conditional probability distribution which is used to compute surprisal values.
`loo`	Should leave-one-out surprisals be computed? For computational reasons, this is only available for `lm` objects.
`...`	Other arguments are ignored.
`approximation`	Character string specifying the method to use in computing the surprisal probabilities. See Details below.
`threshold_probability`	Probability threshold when computing the GPD approximation. This is the probability below which the GPD is fitted. Only used if `approximation = "gpd"`.

Details

The surprisal probabilities may be computed in three different ways.

1. When `approximation = "none"` (the default), the surprisal probabilities are computed using 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. When `approximation = "gdp"`, the surprisal probabilities are computed using a Generalized Pareto Distribution fitted to the most extreme surprisal values (those with probability less than `threshold_probability`). For surprisal probabilities greater 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. When `approximation = "empirical"` (or `"rank"`), the surprisal probability of each observation is estimated using the proportion of observations with greater or equal surprisal values. This is a nonparametric approach that is also insensitive to the distribution used in computing the surprisal values.

Value

A numerical vector containing the surprisals or surprisal probabilities.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 7, https://OTexts.com/weird/density.html.

Hyndman, R J & Frazier, D T (2026) "Anomaly detection using surprisals", https://robjhyndman.com/publications/surprisals.html.

See Also

For specific methods, see `surprisals.numeric()` and `surprisals.lm()`,

Examples

```
# A linear model (i.e., a conditional Gaussian distribution)
lm_of <- lm(waiting ~ duration, data = oldfaithful)
oldfaithful |>
  mutate(
    fscore = surprisals_prob(lm_of),
    prob = surprisals_prob(lm_of, loo = TRUE),
  ) |>
  ggplot(aes(
    x = duration, y = waiting,
    color = prob < 0.01
  )) +
  geom_point()
# A Poisson GLM
glm_breaks <- glm(breaks ~ wool + tension, data = warpbreaks, family = poisson)
warpbreaks |>
  mutate(prob = surprisals_prob(glm_breaks)) |>
  filter(prob < 0.05)
```

surprisals.numeric *Surprisals and surprisal probabilities computed from data*

Description

A surprisal is given by $s = -\log f(y)$ where f is the density or probability mass function of the estimated or assumed distribution, and y is an observation. This is returned by `surprisals()`. A surprisal probability is the probability of a surprisal at least as extreme as s. This is returned by `surprisals_prob()`

Usage

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

## S3 method for class 'matrix'
surprisals(object, distribution = dist_kde(object, ...), loo = FALSE, ...)

## S3 method for class 'data.frame'
surprisals(object, distribution = dist_kde(object, ...), loo = FALSE, ...)

## S3 method for class 'numeric'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  distribution = dist_kde(object, ...),
  loo = FALSE,
  ...
)

## S3 method for class 'matrix'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  distribution = dist_kde(object, ...),
  loo = FALSE,
  ...
)

## S3 method for class 'data.frame'
surprisals_prob(
  object,
  approximation = c("none", "gpd", "empirical", "rank"),
  threshold_probability = 0.1,
  distribution = dist_kde(object, ...),
```

```
  loo = FALSE,
  ...
)
```

Arguments

`object`	A numerical data set (either a vector, matrix, or a data.frame containing only numerical columns).
`distribution`	A distribution object. By default, a kernel density estimate is computed from the data `object`.
`loo`	Should leave-one-out surprisals be computed?
`...`	Other arguments are passed to the appropriate method.
`approximation`	Character string specifying the method to use in computing the surprisal probabilities. See Details below. For a multivariate data set, it needs to be set to either "gpd" or "empirical".
`threshold_probability`	Probability threshold when computing the GPD approximation. This is the probability below which the GPD is fitted. Only used if `approximation = "gpd"`.

Details

The surprisal probabilities may be computed in three different ways.

1. When `approximation = "none"` (the default), the surprisal probabilities are computed using 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. When `approximation = "gdp"`, the surprisal probabilities are computed using a Generalized Pareto Distribution fitted to the most extreme surprisal values (those with probability less than `threshold_probability`). For surprisal probabilities greater 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. When `approximation = "empirical"` (or `"rank"`), the surprisal probability of each observation is estimated using the proportion of observations with greater or equal surprisal values. This is a nonparametric approach that is also insensitive to the distribution used in computing the surprisal values.

Value

A numerical vector containing the surprisals or surprisal probabilities.

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 7, https://OTexts.com/weird/density.html.

Hyndman, R J & Frazier, D T (2026) "Anomaly detection using surprisals", https://robjhyndman.com/publications/surprisals.html.

See Also

dist_kde

Examples

```
# Univariate data
tibble(
  y = c(5, rnorm(49)),
  p_kde = surprisals_prob(y, loo = TRUE),
  p_normal = surprisals_prob(y, distribution = dist_normal()),
  p_zscore = 2 * (1 - pnorm(abs(y)))
)
tibble(
  y = n01$v1,
  prob1 = surprisals_prob(y),
  prob2 = surprisals_prob(y, loo = TRUE),
  prob3 = surprisals_prob(y, distribution = dist_normal()),
  prob4 = surprisals_prob(y, distribution = dist_normal(), approximation = "gpd")
) |>
  arrange(prob1)
# Bivariate data
tibble(
  x = rnorm(50),
  y = c(5, rnorm(49)),
  prob = surprisals_prob(cbind(x, y), approximation = "gpd")
)
oldfaithful |>
  mutate(
    s = surprisals(cbind(duration, waiting), loo = TRUE),
    p = surprisals_prob(cbind(duration, waiting), loo = TRUE, approximation = "gpd")
  ) |>
  arrange(p)
```

---

us_mortality *US weekly mortality*

Description

Weekly deaths and death rates in the USA, by sex and age group, from the second week of 2015 to the last week of 2025. The data are from the Short-Term Mortality Fluctuations (STMF) series of the Human Mortality Database.

Usage

```
us_mortality
```

Format

A data frame with 10314 rows and 6 columns:

Year Year

Week ISO week of the year

Sex "Female", "Male" or "Total"

Age Age group: "0-14", "15-64", "65-74", "75-84", "85+" or "Total"

Deaths Number of deaths

Mortality Weekly death rate: deaths divided by population exposure

Value

Data frame

Source

Human Mortality Database. Max Planck Institute for Demographic Research (Germany), University of California, Berkeley (USA), and French Institute for Demographic Studies (France). https://www.mortality.org

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 12, https://OTexts.com/weird/timeseries.html.

Examples

```
us_mortality |>
  filter(Sex == "Total", Age != "Total") |>
  mutate(time = Year + (Week - 1) / 52) |>
  ggplot(aes(x = time, y = Mortality, colour = Age)) +
  geom_line() +
  scale_y_log10()
```

us_senate *US Senate co-voting networks*

Description

Co-voting networks for the US Senate from the 40th to the 113th Congress (1867 to 2015). Each Congress gives an undirected network in which each node is a senator, and an edge connects two senators if they voted the same way (both yea or both nay) on at least 80% of the bills for which they were both present. The data are taken from Lee, Li and Wilson (2020). Although they state that Independent senators were excluded, a few Independent and minor-party senators are included, coded as Democrat or Republican.

Usage

```
us_senate_edges

us_senate_members
```

Format

`us_senate_edges` is a data frame with 73802 rows and 3 columns:

congress Congress number

from Node number of one senator

to Node number of the other senator

`us_senate_members` is a data frame with 7272 rows and 6 columns:

congress Congress number

node Node number of the senator

name Name of the senator (surname first)

state Two-letter state abbreviation ("USA" for the President)

party "Democrat" or "Republican"

icpsr ICPSR identifier of the senator, as used by Voteview

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 7272 rows and 6 columns.

Details

`us_senate_edges` contains the edges of every network, and `us_senate_members` contains the name, state and party of each senator. Senators are numbered from 1 within each Congress, so the same number refers to different senators in different Congresses; use `icpsr` to follow a senator across Congresses. From the 84th Congress onwards, node 1 is the President, whose announced positions on bills are treated as votes.

Value

Data frame

Source

Lee, J., Li, G., & Wilson, J. D. (2020). Varying-coefficient models for dynamic networks. *Computational Statistics & Data Analysis*, 152, 107052. doi:10.1016/j.csda.2020.107052. Data available from https://github.com/jihuilee/VCERGM.

Names, states and ICPSR identifiers from Lewis, J. B., Poole, K., Rosenthal, H., Boche, A., Rudkin, A., & Sonnet, L. Voteview: Congressional roll-call votes database. https://voteview.com/

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Chapter 16, https://OTexts.com/weird/network.html.

Examples

```
# Number of co-voting edges in each Congress
us_senate_edges |>
  count(congress)
# Party composition of the 100th Congress
us_senate_members |>
  filter(congress == 100) |>
  count(party)
# Congresses in which Patrick Leahy served
us_senate_members |>
  filter(icpsr == 14307)
```

Index

∗ **datasets**
    cricket_batting, 5
    fashion, 8
    fr_mortality, 14
    gun_deaths, 21
    n01, 29
    oldfaithful, 29
    road_deaths, 33
    us_mortality, 44
    us_senate, 46

air_quality *(*fetch_air_quality*)*, 9
augment.Pca, 3

bagplot, *15*
base::scale(), *27, 28*
biplot_projection, 4

chauvenet_anomalies *(*peirce_anomalies*)*, 32
compute.bagplot, *15*
cricket_batting, 5

density_df, 6
density_df(), *16*
dist_kde, 7, *16*, *18*, *44*
dist_mclust, 8
dixon.test, *21*
dixon_anomalies *(*grubbs_anomalies*)*, 20

fashion, 8
fetch_air_quality, 9
fetch_oz_books, 11
fetch_pets2006, 12
fetch_wine_reviews, 13
find_HDoutliers, *34, 35*
fr_mortality, 14

gam, *40*
gg_bagplot, 15
gg_density, 16
gg_hdrboxplot, 17, *24, 25*
ggplot2::geom_point(), *4*, *31*
glm, *40*
glosh, *19*
glosh_scores, 19
grubbs.test, *20, 21*
grubbs_anomalies, 20
gun_deaths, 21

hampel_anomalies, 22
hdbscan, *19*
hdr_regions, 23
hdr_table, *19*, 24, *24*

interaction(), *38*

kde, *7*
kde_bandwidth, *7*, 25
kde_bandwidth(), *7*

lm, *40*
lof, *26, 27*
lof_scores, 26

mclust::Mclust, *8*
mvscale, 27

n01, 29

oldfaithful, 29
outlier_map, 30
oz_books *(*fetch_oz_books*)*, 11

peirce_anomalies, 32
pets2006 *(*fetch_pets2006*)*, 12

road_deaths, 33
robustbase::covMcd(), *28*
robustbase::covOGK(), *28*
robustbase::s_Qn(), *28*
rrcov::PcaClassic(), *3*

rrcov::PcaHubert(), *3*

stats::cov(), *28*
stats::prcomp(), *4*, *30, 31*
stats::sd(), *28*
stray_anomalies, 34
stray_scores, 35
surprisals, *19*, 36
surprisals.data.frame *(*surprisals.numeric*)*, 42
surprisals.factor, 37
surprisals.gam *(*surprisals.lm*)*, 39
surprisals.glm *(*surprisals.lm*)*, 39
surprisals.lm, 39
surprisals.lm(), *37*, *41*
surprisals.matrix *(*surprisals.numeric*)*, 42
surprisals.numeric, 42
surprisals.numeric(), *37*, *41*
surprisals_prob *(*surprisals*)*, 36
surprisals_prob.data.frame *(*surprisals.numeric*)*, 42
surprisals_prob.factor *(*surprisals.factor*)*, 37
surprisals_prob.gam *(*surprisals.lm*)*, 39
surprisals_prob.glm *(*surprisals.lm*)*, 39
surprisals_prob.lm *(*surprisals.lm*)*, 39
surprisals_prob.matrix *(*surprisals.numeric*)*, 42
surprisals_prob.numeric *(*surprisals.numeric*)*, 42

tibble::tibble(), *3*

us_mortality, 44
us_senate, 46
us_senate_edges *(*us_senate*)*, 46
us_senate_members *(*us_senate*)*, 46

wine_reviews *(*fetch_wine_reviews*)*, 13