

Package ‘weird’

May 8, 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://0Texts.com/weird/>>.

All packages needed to run the examples are also loaded.

Version 2.1.0

Depends R (>= 4.1.0)

Imports apIpack, broom, cli (>= 1.0.0), crayon (>= 1.3.4), dbscan, distributional, dplyr (>= 0.7.4), evd, ggplot2 (>= 3.1.1), grDevices, ks, mlpack, mvtnorm, purrr (>= 0.2.4), RANN, rlang, robustbase, rstudioapi (>= 0.7), stray, vctrs

Suggests mgcv, 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

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/Needs/website tidyverse/tidytemplate

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-05-05 07:40:02 UTC

Contents

cricket_batting	2
dist_density	3
dist_kde	4
fetch_wine_reviews	5
fr_mortality	6
gg_bagplot	7
gg_density	8
gg_density_layer	9
gg_hdrboxplot	10
glosh_scores	12
grubbs_anomalies	13
hampel_anomalies	14
hdr_table	15
kde_bandwidth	16
lof_scores	17
mvscale	18
n01	19
oldfaithful	20
peirce_anomalies	21
stray_anomalies	22
stray_scores	23
surprisals	24
surprisals.lm	25
surprisals.numeric	28
Index	31

cricket_batting	<i>Cricket batting data for international test players</i>
-----------------	--

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))
```

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.

Value

A distributional object of class `dist_density`.

Examples

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

dist_kde	<i>Create distributional object based on a kernel density estimate</i>
----------	--

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", "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 <code>kde_bandwidth</code> function is used.
H	Bandwidth matrix for multivariate distribution. If NULL, the <code>kde_bandwidth</code> function is used.
method	The method of bandwidth estimation to use. See <code>kde_bandwidth()</code> for details. Ignored if h or H are specified.
...	Other arguments are passed to <code>kde</code> .

Value

A distributional object of class `dist_kde`.

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.7 and 3.9, <https://OTexts.com/weird/>.

Examples

```
dist_kde(c(rnorm(200), rnorm(100, 5)))  
dist_kde(cbind(rnorm(200), rnorm(200, 5)))
```

fetch_wine_reviews	<i>Wine prices and points</i>
--------------------	-------------------------------

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://www.kaggle.com/>

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 1.4, <https://0Texts.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", Section 1.4, <https://0Texts.com/weird/>.

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

<code>data</code>	A data frame or matrix containing the data.
<code>var1</code>	The name of the first variable to plot (a bare expression).
<code>var2</code>	The name of the second variable to plot (a bare expression).
<code>color</code>	The base color to use for the median. Other colors are generated as a mixture of color with white.
<code>show_points</code>	A logical argument indicating if a regular bagplot is required (FALSE), or if a scatterplot in the same colors is required (TRUE).
<code>...</code>	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	<i>Produce ggplot of densities from distributional objects in 1 or 2 dimensions</i>
------------	---

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 (using a GPD approximation) are shown in black.
colors	Color palette to use. If there are more than <code>length(colors)</code> distributions, they are recycled. Default is the Okabe-Ito color palette.
alpha	Transparency of points. Ignored if <code>show_points</code> is FALSE. Defaults to <code>min(1, 500/n)</code> , where <code>n</code> is the number of observations plotted.
jitter	For univariate distributions, when <code>jitter</code> is TRUE and <code>show_points</code> 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 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_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

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

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,  
  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 (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.
ngrid	Number of grid points to use for the density function.
...	Other arguments passed to <code>dist_kde</code> .

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 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

<code>y</code>	Numerical matrix or vector of data
<code>k</code>	Minimum cluster size. Default: 5.
<code>...</code>	Additional arguments passed to <code>dbSCAN</code> : 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	<i>Statistical tests for anomalies using Grubbs' test and Dixon's test</i>
------------------	--

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 $\text{logit}(\alpha)$ 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))
```

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 $2 * \text{bandwidth} + 1$. 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 $\text{MAD} * 1.4826$, 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 9.2, <https://OTexts.com/weird/>.

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)
```

Arguments

object	Distributional object such as that returned by <code>dist_kde()</code>
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 2.5, 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, 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", "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), 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 2.7 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

<code>y</code>	Numerical matrix or vector of data
<code>k</code>	Number of neighbours to include. Default: 5.
<code>...</code>	Additional arguments passed to <code>dbSCAN::lof</code>

Value

Numerical vector containing LOF values

Author(s)

Rob J Hyndman

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 7.3, <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, and uses robust estimates of center, scale and covariance by default. The centers are removed using medians, the scale function for univariate data is `s_Qn`, the covariance matrix for multivariate data is estimated using a robust OGK 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::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 <code>cov = robustbase::covOGK()</code> , it is passed as the <code>sigmanu</code> 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

References

Hyndman, R J (2026) "That's weird: Anomaly detection using R", Section 2.6, 3.6 and 3.7.

See Also

[base::scale\(\)](#), [stats::sd\(\)](#), [stats::cov\(\)](#), [robustbase::covOGK\(\)](#), [robustbase::s_Qn\(\)](#)

Examples

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

peirce_anomalies	<i>Anomalies according to Peirce's and Chauvenet's criteria</i>
------------------	---

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))
```

stray_anomalies	<i>Stray anomalies</i>
-----------------	------------------------

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	<i>Stray scores</i>
--------------	---------------------

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)
threshold <- stray::find_threshold(scores, alpha = 0.01, outtail = "max", p = 0.5, tn = 50)
which(scores > threshold)
```

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

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

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 = "gpd"`, 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.

- 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 6, <https://0texts.com/weird/>.

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.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

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

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 = "gpd"`, 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 6, <https://OTexts.com/weird/>.

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 = "gpd"`, 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 6, <https://OTexts.com/weird/>.

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)
```

Index

- * **datasets**
 - cricket_batting, [2](#)
 - fr_mortality, [6](#)
 - n01, [19](#)
 - oldfaithful, [20](#)
- bagplot, [7](#)
- base::scale(), [18](#), [19](#)
- chauenet_anomalies (peirce_anomalies), [21](#)
- compute.bagplot, [7](#)
- cricket_batting, [2](#)
- dist_density, [3](#)
- dist_kde, [4](#), [8](#), [9](#), [11](#), [30](#)
- dixon.test, [14](#)
- dixon_anomalies (grubbs_anomalies), [13](#)
- fetch_wine_reviews, [5](#)
- find_HDoutliers, [22](#), [23](#)
- fr_mortality, [6](#)
- gam, [26](#)
- geom_line, [9](#)
- gg_bagplot, [7](#)
- gg_density, [8](#)
- gg_density_layer, [9](#)
- gg_hdrboxplot, [10](#), [16](#)
- glm, [26](#)
- glosh, [12](#)
- glosh_scores, [12](#)
- grubbs.test, [13](#), [14](#)
- grubbs_anomalies, [13](#)
- hampel_anomalies, [14](#)
- hdbscan, [12](#)
- hdr_table, [11](#), [15](#)
- kde, [4](#)
- kde_bandwidth, [4](#), [16](#)
- kde_bandwidth(), [4](#)
- lm, [26](#)
- lof, [17](#), [18](#)
- lof_scores, [17](#)
- mvscale, [18](#)
- n01, [19](#)
- oldfaithful, [20](#)
- peirce_anomalies, [21](#)
- robustbase::covOGK(), [19](#)
- robustbase::s_Qn(), [19](#)
- stats::cov(), [19](#)
- stats::sd(), [19](#)
- stray_anomalies, [22](#)
- stray_scores, [23](#)
- surprisals, [11](#), [24](#)
- surprisals.data.frame
 - (surprisals.numeric), [28](#)
- surprisals.gam (surprisals.lm), [25](#)
- surprisals.glm (surprisals.lm), [25](#)
- surprisals.lm, [25](#)
- surprisals.lm(), [25](#), [27](#)
- surprisals.matrix (surprisals.numeric), [28](#)
- surprisals.numeric, [28](#)
- surprisals.numeric(), [25](#), [27](#)
- surprisals_prob (surprisals), [24](#)
- surprisals_prob.data.frame
 - (surprisals.numeric), [28](#)
- surprisals_prob.gam (surprisals.lm), [25](#)
- surprisals_prob.glm (surprisals.lm), [25](#)
- surprisals_prob.lm (surprisals.lm), [25](#)
- surprisals_prob.matrix
 - (surprisals.numeric), [28](#)

surprisals_prob.numeric
 (surprisals.numeric), [28](#)

wine_reviews (fetch_wine_reviews), [5](#)