# Package: lookout (via r-universe)

October 18, 2026

**Type** Package

**Title** Leave One Out Kernel Density Estimates for Outlier Detection

**Version** 2.0.2.9000

**Maintainer** Sevvandi Kandanaarachchi <sevvandik@gmail.com>

**Description** Outlier detection using leave-one-out kernel density estimates and extreme value theory. The bandwidth for the kernel density estimates is computed from the edge lengths of the Euclidean minimum spanning tree of the data. Using the peak-over-threshold method, a generalized Pareto distribution is fitted to the negative log of the leave-one-out kernel density estimates to identify outliers. See Kandanaarachchi and Hyndman (2022) <doi:10.1080/10618600.2021.2000425> and Hyndman, Kandanaarachchi and Turner (2026) <doi:10.48550/arXiv.2603.22636>.

**License** GPL-3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**BugReports** https://github.com/sevvandi/lookout/issues

**Depends** R (>= 4.1.0)

**Imports** dbscan, evd, ggplot2, mlpack, robustbase, stats, tidyr

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**URL** https://sevvandi.github.io/lookout/, https://github.com/sevvandi/lookout

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.1.0

**Config/pak/sysreqs** libicu-dev

**Repository** https://sevvandi.r-universe.dev

**Date/Publication** 2026-09-18 03:52:15 UTC

**RemoteUrl** https://github.com/sevvandi/lookout

**RemoteRef** HEAD

**RemoteSha** 3ae6ac40241afceed7dea5f4d82a6dee38ca6f00

# Contents

autoplot.lookoutliers . . . . . 2
autoplot.persistingoutliers . . . . . 3
find_tda_bw . . . . . 4
lookout . . . . . 5
lookout_ts . . . . . 8
mvscale . . . . . 9
persisting_outliers . . . . . 10

**Index** **13**

---

`autoplot.lookoutliers` *Plots outliers identified by lookout algorithm.*

---

## Description

Scatterplot of two columns from the data set with outliers highlighted.

## Usage

```
## S3 method for class 'lookoutliers'
autoplot(object, columns = 1:2, ...)
```

## Arguments

| | |
|---|---|
| `object` | The output of the function `lookout`. |
| `columns` | Which columns of the original data to plot (specified as either numbers or strings) |
| `...` | Other arguments currently ignored. |

## Value

A ggplot object.

## Examples

```
X <- rbind(
  data.frame(
    x = rnorm(500),
    y = rnorm(500)
  ),
  data.frame(
    x = rnorm(5, mean = 10, sd = 0.2),
    y = rnorm(5, mean = 10, sd = 0.2)
  )
)
lo <- lookout(X)
autoplot(lo)
```

`autoplot.persistingoutliers` *Plots outlier persistence for a range of significance levels.*

## Description

This function plots outlier persistence for a range of significance levels using the algorithm lookout, an outlier detection method that uses leave-one-out kernel density estimates and generalized Pareto distributions to find outliers.

## Usage

```
## S3 method for class 'persistingoutliers'
autoplot(object, alpha = object$alpha, ...)
```

## Arguments

| | |
|---|---|
| `object` | The output of the function `persisting_outliers`. |
| `alpha` | The significance levels to plot. |
| `...` | Other arguments currently ignored. |

## Value

A ggplot object.

## Examples

```
X <- rbind(
  data.frame(
    x = rnorm(500),
    y = rnorm(500)
  ),
  data.frame(
    x = rnorm(5, mean = 10, sd = 0.2),
    y = rnorm(5, mean = 10, sd = 0.2)
  )
)
plot(X, pch = 19)
outliers <- persisting_outliers(X, scale = FALSE)
autoplot(outliers)
```

`find_tda_bw` *Computes the bandwidth for lookout from the minimum spanning tree.*

## Description

This function computes the bandwidth used by `lookout()` for its kernel density estimates, from the edge lengths of the Euclidean minimum spanning tree of the data.

## Usage

```
find_tda_bw(X, fast = NULL, gamma = 0.98, use_differences = FALSE)
```

## Arguments

| | |
|---|---|
| `X` | The numerical input data in a data.frame, matrix or tibble format. |
| `fast` | Deprecated and ignored. The bandwidth is always computed using all of `X`. |
| `gamma` | The quantile of the minimum spanning tree edge lengths to return. Default is `0.98`. Ignored when `use_differences = TRUE`. |
| `use_differences` | If `TRUE`, the value returned is the lower end of the largest gap between consecutive minimum spanning tree edge lengths among those at or above their median. If `FALSE` (the default), the `gamma` quantile of the edge lengths is returned. |

## Details

The Euclidean minimum spanning tree of the rows of `X` is computed with `emst()`, giving `NROW(X) - 1` edge lengths in increasing order. Each edge length is the distance at which two clusters merge under single-linkage clustering, or equivalently the death radius of a connected component in the Rips filtration of the data, which is why the function is named after topological data analysis.

By default (`use_differences = FALSE`), the value returned is the `gamma` quantile of the edge lengths, computed with `quantile()` using `type = 8`. With `use_differences = TRUE`, only the edge lengths at or above their median are kept, and the value returned is the lower of the two consecutive edge lengths with the largest gap between them. This is the rule of Kandanaarachchi and Hyndman (2022), used by `lookout()` when `old_version = TRUE`.

The value returned is on the scale of `X` and is not itself the support radius of the kernel. `lookout()` multiplies it by `sqrt(m + 4)`, where `m = NCOL(X)`, to obtain the support radius of the Epanechnikov kernel. A spherically symmetric Epanechnikov kernel with support radius `h` in `m` dimensions has standard deviation `h / sqrt(m + 4)` in each coordinate, so the kernel used by `lookout()` has per-coordinate standard deviation equal to the value returned here. Note that `lookout()` applies this function to the scaled data when `scale = TRUE`.

The minimum spanning tree is always computed on all of `X`.

## Value

A single number: the bandwidth on the scale of `X`, before multiplication by `sqrt(NCOL(X) + 4)`.

## References

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

Hyndman, RJ, Kandanaarachchi, S, and Turner, K (2026) Lookout 2: Anomaly detection via leave-one-out kernel density estimation, arXiv:2603.22636. https://robjhyndman.com/publications/lookout2.html

## Examples

```
X <- rbind(
  data.frame(
    x = rnorm(500),
    y = rnorm(500)
  ),
  data.frame(
    x = rnorm(5, mean = 10, sd = 0.2),
    y = rnorm(5, mean = 10, sd = 0.2)
  )
)
find_tda_bw(X)
```

---

`lookout` *Identifies outliers using the algorithm lookout.*

---

## Description

This function identifies outliers using the algorithm lookout, an outlier detection method that uses leave-one-out kernel density estimates and generalized Pareto distributions to find outliers.

## Usage

```
lookout(
  X,
  alpha = 0.01,
  beta = 0.9,
  gamma = 0.98,
  bw = NULL,
  gpd = NULL,
  scale = TRUE,
  fast = NROW(X) > 10000,
  old_version = FALSE
)
```

### Arguments

| | |
|---|---|
| `X` | The numerical input data in a data.frame, matrix or tibble format. |
| `alpha` | The level of significance. Default is `0.01`. So there is a 1/100 chance of any point being falsely classified as an outlier. |
| `beta` | The quantile threshold used in the GPD estimation. Default is `0.90`. To ensure there is enough data available, values greater than 0.90 are set to 0.90. |
| `gamma` | The quantile of the minimum spanning tree edge lengths used to compute the bandwidth. Default is `0.98`. Ignored if `bw` is provided, and ignored when `old_version = TRUE`, where the largest gap between consecutive edge lengths is used instead. See Details. |
| `bw` | The support radius of the Epanechnikov kernel, on the scale of the data after any scaling. If `NULL` (default), it is computed from the minimum spanning tree of the data as described in Details. |
| `gpd` | Generalized Pareto distribution parameters. If `NULL` (the default), these are estimated from the data. |
| `scale` | If `TRUE` (the default), the data are scaled before the bandwidth and density estimates are computed: with `mvscale()` when `old_version = FALSE`, so that the columns are approximately uncorrelated with unit scale, or by scaling each column to the range `[0, 1]` when `old_version = TRUE`. |
| `fast` | If `TRUE`, each kernel density estimate is a sum over the `k` nearest neighbours of the point only, including the point itself, where `k = min(max(ceiling(n / 200), 100), n, 500)` and `n = NROW(X)`; so `k` is between 100 and 500, and equals 100 whenever `100 <= n <= 20000`. Wherever more than `k - 1` other observations lie inside the kernel support, which is typical in the bulk of the data in three or more dimensions, the sum is truncated and the estimate is lower than the exact one. Sparse observations, which are the candidates for outliers, have fewer than `k` neighbours inside the support and their estimates are unchanged, although the GPD threshold and fit can still differ. If `FALSE`, each kernel is summed over exactly the observations inside its support, found with a fixed-radius search (`frNN()`). The time and memory of this exact computation are proportional to the number of pairs of observations within `bw` of each other. In two dimensions this is usually modest, but in three or more dimensions the kernel support in the bulk of the data can contain thousands of observations, so the exact computation is still of order `n^2` in the worst case and `fast = TRUE` remains the practical choice for large `n`. Default is `TRUE` when `NROW(X) > 10000`. The bandwidth calculation always uses all of the data. |
| `old_version` | If `TRUE`, the algorithm of Kandanaarachchi and Hyndman (2022) is used. Default is `FALSE`, giving the algorithm of Hyndman, Kandanaarachchi and Turner (2026). See Details. |

### Details

The algorithm has three steps.

**Scaling.** When `scale = TRUE`, the data are first scaled with `mvscale()`: each column is centred at its median, and the data are rotated and scaled using the Cholesky factor of the inverse of a robust (MCD) covariance estimate, so that the columns are approximately uncorrelated with unit scale.

**Leave-one-out density estimates.** A kernel density estimate is computed at each observation using a spherically symmetric Epanechnikov kernel with support radius `bw`, and the contribution of the observation itself is then removed to give a leave-one-out estimate. When `bw = NULL`, the bandwidth is computed from the Euclidean minimum spanning tree of the (scaled) data by `find_tda_bw()`: the `gamma` quantile (type 8) of the tree's edge lengths is multiplied by `sqrt(m + 4)`, where `m = NCOL(X)`. A spherically symmetric Epanechnikov kernel with support radius `h` in `m` dimensions has standard deviation `h / sqrt(m + 4)` in each coordinate, so this scaling makes the kernel's per-coordinate standard deviation equal to the quantile. When `fast = TRUE`, each kernel is summed over the `k` nearest neighbours of the point only; see the `fast` argument.

**Extreme value model.** The negative logarithms of the density estimates (the surprisals) above their `beta` quantile are modelled with a generalized Pareto distribution (GPD), fitted by maximum likelihood using `fpot()`. Because the surprisals are bounded, the shape parameter is constrained to be at most zero: if the unconstrained estimate is positive, the GPD is refitted with the shape fixed at zero. The fitted GPD gives, for each observation, the probability of a leave-one-out surprisal at least as large as the one observed, multiplied by `1 - beta`. Observations whose probability is below `alpha` are declared outliers.

Setting `old_version = TRUE` gives the algorithm of Kandanaarachchi and Hyndman (2022) instead. It differs in three places: the data are scaled so that each column lies in `[0, 1]`, rather than with `mvscale()`; the bandwidth is the lower end of the largest gap between consecutive minimum spanning tree edge lengths among those at or above their median, rather than the `gamma` quantile, again multiplied by `sqrt(m + 4)`; and the GPD shape parameter is not constrained.

## Value

A list with the following components:

| | |
|---|---|
| `data` | The input data `X`, before any scaling. |
| `outliers` | The set of outliers. |
| `outlier_probability` | The GPD probability of the data. |
| `outlier_scores` | The outlier scores of the data. |
| `bandwidth` | The support radius of the Epanechnikov kernel: either `bw`, or the value computed from the minimum spanning tree multiplied by `sqrt(NCOL(X) + 4)`. |
| `kde` | The kernel density estimate values. |
| `lookde` | The leave-one-out kde values. |
| `gpd` | The fitted GPD parameters. |
| `call` | The matched call. |

## References

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

Hyndman, RJ, Kandanaarachchi, S, and Turner, K (2026) Lookout 2: Anomaly detection via leave-one-out kernel density estimation, arXiv:2603.22636. https://robjhyndman.com/publications/lookout2.html

## Examples

```
X <- rbind(
  data.frame(
    x = rnorm(500),
    y = rnorm(500)
  ),
  data.frame(
    x = rnorm(5, mean = 10, sd = 0.2),
    y = rnorm(5, mean = 10, sd = 0.2)
  )
)
lo <- lookout(X)
lo
autoplot(lo)
```

---

`lookout_ts` *Identifies outliers in univariate time series using the algorithm lookout.*

---

## Description

This is the time series implementation of lookout which identifies outliers in the double differenced time series.

## Usage

```
lookout_ts(x, scale = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | The input univariate time series. |
| `scale` | If `TRUE` (the default), the data are scaled before the bandwidth and density estimates are computed: with `mvscale()` when `old_version = FALSE`, so that the columns are approximately uncorrelated with unit scale, or by scaling each column to the range `[0, 1]` when `old_version = TRUE`. |
| `...` | Other arguments are passed to `lookout`. |

## Value

A lookout object.

## See Also

`lookout`

## Examples

```
set.seed(1)
x <- arima.sim(list(order = c(1, 1, 0), ar = 0.8), n = 200)
x[50] <- x[50] + 10
plot(x)
lo <- lookout_ts(x)
lo
```

---

`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 3.7.

### See Also

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

### Examples

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

---

## persisting_outliers *Computes outlier persistence for a range of significance values.*

---

### Description

This function computes outlier persistence for a range of significance values, using the algorithm lookout, an outlier detection method that uses leave-one-out kernel density estimates and generalized Pareto distributions to find outliers.

## Usage

```
persisting_outliers(
  X,
  alpha = seq(0.01, 0.1, by = 0.01),
  st_qq = 0.9,
  scale = TRUE,
  num_steps = 20,
  old_version = FALSE
)
```

## Arguments

| | |
|---|---|
| X | The input data in a matrix, data.frame, or tibble format. All columns should be numeric. |
| alpha | Grid of significance levels. |
| st_qq | The quantile of the minimum spanning tree edge lengths at which the bandwidth sequence starts. The sequence ends at the largest edge length. Both are multiplied by `sqrt(NCOL(X) + 4)` to give the support radius of the Epanechnikov kernel; see `find_tda_bw()`. |
| scale | If `TRUE`, the data is scaled. Default is `TRUE`. Which scaling method is used depends on the `old_version` parameter. See `lookout` for details. |
| num_steps | The length of the bandwidth sequence. |
| old_version | Logical indicator of which version of the algorithm to use. |

## Value

A list with the following components:

| | |
|---|---|
| out | A 3D array of `N` x `num_steps` x `num_alpha` where `N` denotes the number of observations, `num_steps` denote the length of the bandwidth sequence, and `num_alpha` denotes the number of significance levels. This is a binary array and the entries are set to 1 if that observation is an outlier for that particular bandwidth and significance level. |
| bw | The set of bandwidth values. |
| gpdparas | The GPD parameters used. |
| lookoutbw | The bandwidth used for the GPD fit: the lower end of the largest gap between consecutive minimum spanning tree edge lengths among those at or above their median (the rule used by `lookout()` when `old_version = TRUE`), multiplied by `sqrt(NCOL(X) + 4)`. |

## Examples

```
X <- rbind(
  data.frame(
    x = rnorm(500),
    y = rnorm(500)
  ),
```

```
  data.frame(
    x = rnorm(5, mean = 10, sd = 0.2),
    y = rnorm(5, mean = 10, sd = 0.2)
  )
)
plot(X, pch = 19)
outliers <- persisting_outliers(X, scale = FALSE)
outliers
autoplot(outliers)
```

# Index

autoplot.lookoutliers, 2
autoplot.persistingoutliers, 3

base::scale(), *9, 10*

emst, *4*

find_tda_bw, 4, *7*, *11*
fpot, *7*
frNN, *6*

lookout, *4*, 5, *8*, *11*
lookout_ts, 8

mvscale, *6–8*, 9

persisting_outliers, 10

quantile, *4*

robustbase::covMcd(), *10*
robustbase::covOGK(), *10*
robustbase::s_Qn(), *10*

stats::cov(), *10*
stats::sd(), *10*