Package 'quickOutlier'

February 13, 2026

Title Detect and Treat Outliers in Data Mining

Version 0.1.5

Description Implements a suite of tools for outlier detection and treatment in data mining. It includes univariate methods (Z-score, Interquartile Range), multivariate detection using Mahalanobis distance, and density-based detection (Local Outlier Factor) via the 'dbscan' package. It also provides functions for visualization using 'ggplot2' and data cleaning via Winsorization.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports dbscan, ggplot2, isotree, plotly, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/daniellop1/quickOutlier

BugReports https://github.com/daniellop1/quickOutlier/issues

NeedsCompilation no

Author Daniel López Pérez [aut, cre] (ORCID: <https://orcid.org/0009-0008-1695-9837>)

Maintainer Daniel López Pérez <dlopez350@icloud.com>

Repository CRAN

Date/Publication 2026-02-13 19:40:02 UTC

Contents

`detect_categorical_outliers`
Detect Rare Categories (Categorical Outliers)

Description

Identifies categories in a character or factor vector that appear less frequently than a specified threshold.

Usage

```
detect_categorical_outliers(data, min_freq = 0.01)
```

Arguments

`data`	A vector (character or factor).
`min_freq`	Numeric. The minimum percentage (0 to 1) required to be considered normal. Defaults to 0.01 (1 percent).

Details

The function calculates the relative frequency of each unique level. If the frequency is below `min_freq`, the category is flagged as an outlier.

Value

A data frame summarizing the categories:

`Category`	The name of the level.
`Count`	Absolute frequency.
`Frequency`	Relative frequency.
`Is_Outlier`	Logical flag.

Examples

```
cities <- c(rep("Madrid", 10), "Barcalona")
detect_categorical_outliers(cities, min_freq = 0.1)
```

detect_density *Detect Density-Based Anomalies (LOF)*

Description

Uses the Local Outlier Factor (LOF) algorithm to identify anomalies based on local density. It is useful for detecting outliers in multi-dimensional data that Z-score misses.

Usage

```
detect_density(data, k = 5, threshold = 1.5)
```

Arguments

data	A data frame (only numeric columns will be used).
k	Integer. The number of neighbors to consider. Defaults to 5.
threshold	Numeric. The LOF score cutoff. Values > 1 indicate potential outliers. Defaults to 1.5.

Value

A data frame with the outliers and their LOF score.

Examples

```
df <- data.frame(x = c(rnorm(50), 5), y = c(rnorm(50), 5))
detect_density(df, k = 5)
```

detect_iforest *Detect Outliers using Isolation Forest (Machine Learning)*

Description

This function applies the Isolation Forest algorithm to detect anomalies in high-dimensional or complex datasets. Unlike statistical methods that measure distance from a mean, Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Usage

```
detect_iforest(data, ntrees = 100, contamination = 0.05)
```

Arguments

`data`	A data frame containing at least one numeric column. Non-numeric columns are ignored.
`ntrees`	Integer. The number of trees to grow in the forest. Defaults to 100. Increasing this number improves accuracy but increases computation time.
`contamination`	Numeric (0 to 0.5). The expected proportion of outliers in the dataset. Used to calculate the threshold for the binary `Is_Outlier` flag. Defaults to 0.05 (5%).

Details

Recursive partitioning can be represented by a tree structure, and the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. Random trees produce shorter path lengths for anomalies, as they are essentially "fewer" and "different" from normal observations.

The function relies on the efficient `isotree` package for computation.

Value

A data frame with the original columns plus:

`If_Score`	Numeric score between 0 and 1. Higher values indicate higher anomaly likelihood.
`Is_Outlier`	Logical flag. TRUE if the score exceeds the quantile defined by the contamination rate.

Note

This function requires the `isotree` package.

See Also

`isolation.forest`

Examples

```
# Example: Detect anomalies in a generated dataset
df <- data.frame(x = c(rnorm(100), 1000), y = c(rnorm(100), 1000))
result <- detect_iforest(df, ntrees = 50, contamination = 0.02)
tail(result)
```

`detect_multivariate` *Detect Multivariate Anomalies (Mahalanobis Distance)*

Description

Identifies outliers based on the relationship between multiple variables using Mahalanobis Distance. This is useful when individual values are normal, but their combination is anomalous (e.g., high weight for low height).

Usage

```
detect_multivariate(data, columns, confidence_level = 0.99)
```

Arguments

`data`	A data frame.
`columns`	Vector of column names to analyze (must be numeric).
`confidence_level`	Numeric (0 to 1). The confidence cutoff for the Chi-square distribution. Defaults to 0.99 (99%).

Value

A data frame with the multivariate outliers and their Mahalanobis distance.

Examples

```
# Generate dataset (n=50) with strong correlation
df <- data.frame(x = rnorm(50), y = rnorm(50))
df$y <- df$x * 2 + rnorm(50, sd = 0.5) # y depends on x

# Add an anomaly: normal x, but impossible y
anomaly <- data.frame(x = 0, y = 10)
df <- rbind(df, anomaly)

# Detect
detect_multivariate(df, columns = c("x", "y"))
```

`detect_outliers` *Detect Anomalies in a Data Frame*

Description

This function identifies rows containing outliers in a specific numeric column. It supports two methods:

- **zscore**: Based on the standard deviation (statistical approach). Best for normal distributions.
- **iqr**: Based on the Interquartile Range (robust approach). Best for data with skewness or extreme outliers.

Usage

```
detect_outliers(data, column, method = "zscore", threshold = 3)
```

Arguments

`data`	A data frame containing the data to analyze.
`column`	A string specifying the name of the numeric column to analyze.
`method`	A character string. "zscore" or "iqr". Defaults to "zscore".
`threshold`	A numeric value. The cutoff limit. Defaults to 3 for "zscore" and 1.5 for "iqr".

Value

A data frame containing only the rows considered outliers, with an additional column displaying the calculated score or bounds.

Examples

```
# Example with a clear outlier
df <- data.frame(
  id = 1:6,
  value = c(10, 12, 11, 10, 500, 11)
)

# Detect using IQR (Robust)
detect_outliers(df, column = "value", method = "iqr")

# Detect using Z-Score
detect_outliers(df, column = "value", method = "zscore")
```

`detect_ts_outliers` *Detect Anomalies in Time Series using STL Decomposition*

Description

Performs a seasonal-trend decomposition using Loess (STL) to separate the time series into three components: Trend, Seasonality, and Remainder (Noise). Outliers are then detected in the Remainder component using the Interquartile Range (IQR) method.

Usage

```
detect_ts_outliers(data, frequency = 12)
```

Arguments

`data`	A numeric vector representing the time series values.
`frequency`	Integer. The number of observations per cycle (e.g., 12 for monthly data, 7 for daily data).

Details

This method is superior to simple thresholding for time series because it accounts for:

- **Seasonality:** Repeating patterns (e.g., higher sales in December).
- **Trend:** Long-term increase or decrease.

An observation is flagged as an outlier only if it is unusual *after* accounting for these normal temporal patterns.

Value

A data frame containing:

`Original`	The original values.
`Trend`	The extracted long-term trend component.
`Seasonal`	The extracted seasonal component.
`Remainder`	The remaining noise after removing trend and seasonality.
`Is_Outlier`	Logical flag. TRUE if the Remainder value is an outlier based on IQR (3 * IQR).

Examples

```
# Example: Synthetic monthly data with a clear anomaly
sales <- c(sin(seq(1, 20, 0.5)) * 10 + 50) # Normal pattern
sales[10] <- 200 # Inject outlier
result <- detect_ts_outliers(sales, frequency = 12)
subset(result, Is_Outlier == TRUE)
```

`diagnose_influence` *Diagnose Influential Points in Linear Models (Cook's Distance)*

Description

Fits a linear model between two variables and calculates Cook's Distance to identify influential points. An influential point is an outlier that specifically affects the slope of the regression line.

Usage

```
diagnose_influence(data, target, predictor, cutoff = NULL)
```

Arguments

`data`	A data frame containing the variables.
`target`	Character. The name of the dependent variable (Y).
`predictor`	Character. The name of the independent variable (X).
`cutoff`	Numeric (Optional). A custom threshold for Cook's Distance. If NULL, it defaults to 4/n.

Details

Cook's distance (D_i) measures the effect of deleting a given observation. Points with a large D_i are considered to have high leverage and influence.

The default threshold for detection is calculated as:

$$Threshold = \frac{4}{n}$$

Where n is the number of observations. This is a standard rule of thumb in regression diagnostics.

Value

A data frame with the original data plus:

`Cooks_Dist`	The calculated Cook's distance for each point.
`Is_Influential`	Logical flag. TRUE if Cooks_Dist > cutoff.

Examples

```
# Example: A point that pulls the regression line
df <- mtcars
# Artificially create a leverage point
df[1, "wt"] <- 10
df[1, "mpg"] <- 50
result <- diagnose_influence(df, "mpg", "wt")
head(result)
```

plot_interactive *Create an Interactive Outlier Plot*

Description

Generates an interactive scatter plot using 'plotly'. Outliers detected by the specified method are highlighted in red.

Usage

```
plot_interactive(data, x_col, y_col, confidence_level = 0.95)
```

Arguments

data	A data frame.
x_col	The name of the numeric column for the X-axis.
y_col	The name of the numeric column for the Y-axis.
confidence_level	Numeric. Threshold for detection (0.95, 0.99). Lower values make detection stricter.

Value

A plotly object.

plot_outliers *Plot Outliers with ggplot2*

Description

Visualizes the distribution of a variable and highlights detected outliers in red. It combines a boxplot (for context) and jittered points (for individual data visibility).

Usage

```
plot_outliers(data, column, method = "zscore", threshold = 3)
```

Arguments

data	A data frame.
column	The name of the numeric column to plot.
method	"zscore" or "iqr". Defaults to "zscore".
threshold	Numeric. Defaults to 3 for zscore, 1.5 for IQR.

Value

A ggplot object. You can add more layers to it using +.

Examples

```
library(ggplot2)
df <- data.frame(val = c(rnorm(50), 10)) # 50 normal points and one outlier
plot_outliers(df, "val", method = "iqr")
```

---

scan_data *Scan Entire Dataset for Outliers*

Description

Iterates through all numeric columns in the dataset and provides a summary table of outliers found.

Usage

```
scan_data(data, method = "iqr")
```

Arguments

data	A data frame.
method	"iqr" or "zscore". Defaults to "iqr".

Value

A summary data frame with columns: Column, Outlier_Count, and Percentage.

Examples

```
df <- data.frame(
  a = c(1:10, 100),
  b = c(1:10, 1)
)
scan_data(df, method = "iqr")
```

`treat_outliers` *Treat Outliers (Winsorization/Capping)*

Description

Instead of removing outliers, this function replaces extreme values with the calculated upper and lower boundaries (caps). This technique is often called "Winsorization".

Usage

```
treat_outliers(data, column, method = "iqr", threshold = 1.5)
```

Arguments

`data`	A data frame.
`column`	The numeric column to treat.
`method`	"iqr" or "zscore".
`threshold`	Numeric (1.5 for IQR, 3 for zscore).

Value

A data frame with the modified column values.

Examples

```
# Example: 100 is an outlier
df <- data.frame(val = c(1, 2, 3, 2, 1, 100))

# The 100 will be replaced by the maximum allowed IQR value
clean_df <- treat_outliers(df, "val", method = "iqr")
print(clean_df$val)
```

Index