Package 'quickOutlier'

December 19, 2025

Title Detect and Treat Outliers in Data Mining

Version 0.1.0

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, 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]

Maintainer Daniel López Pérez <dlopez350@icloud.com>

Repository CRAN

Date/Publication 2025-12-19 15:00:02 UTC

Contents

`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_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")
```

`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