Package: cyclicwave (via r-universe)

July 3, 2026

Type Package

Title Cyclic Wave Analysis for Time-Series Clustering

Version 0.1.0

Description A modular toolkit for feature extraction and density-based clustering of time-series data. It provides classical statistical, discrete wavelet, Hilbert-based phase, and circular statistical features. The Hilbert-based phase representation can support the analysis of periodic patterns, phase relationships, and circular behavior in time-series data. The package supports DBSCAN and OPTICS clustering, cluster evaluation, visualization, data preparation, and comparison of multiple feature extraction and clustering combinations. Methods are described in Karakaya and Purutcuoglu (2026) <doi:10.15672/hujms.1821412> and Karakaya et al. (2026) <doi:10.1007/978-3-032-17020-0_27>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports stats, utils, dbscan, gsignal, waveslim, MASS, e1071, ggplot2

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

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Depends R (>= 3.5.0)

VignetteBuilder knitr

NeedsCompilation no

Author Şule Şevval Karakaya [aut, cre], Ahmet Bursalı [aut], Vilda Purutçuoğlu [aut]

Maintainer Şule Şevval Karakaya <sule.karakaya@metu.edu.tr>

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

Date/Publication 2026-07-03 12:00:08 UTC

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

RemoteRef HEAD

RemoteSha b53ff7f579ae08763985039e942511a9a74803df

Contents

a_star_statistic *A-Star Statistic*

Description

Calculates an observation-level circular distance measure using the shortest angular separation between pairs of angles.

Usage

```
a_star_statistic(theta)
```

Arguments

theta	Numeric vector containing angles in radians.

Value

Maximum normalized angular distance value.

a_statistic *A Statistic*

Description

Calculates an observation-level circular distance measure based on pairwise cosine differences.

Usage

```
a_statistic(theta)
```

Arguments

theta	Numeric vector containing angles in radians.

Value

Maximum normalized distance value.

analytic_signal *Analytic Signal*

Description

Calculates the complex analytic signal of a numeric vector or matrix using the Hilbert transform.

Usage

```
analytic_signal(X, axis = c("time", "feature"))
```

Arguments

X	Numeric vector, matrix, or data frame.
axis	Direction in which the Hilbert transform is applied. Available options are "time" and "feature".

Value

A complex matrix with the same dimensions as the input.

best_map *Match Predicted Cluster Labels*

Description

Matches each predicted cluster label to the true label with which it has the largest overlap.

Usage

```
best_map(true_labels, pred_labels)
```

Arguments

true_labels	Vector containing the known class labels.
pred_labels	Vector containing the predicted cluster labels.

Value

A numeric vector containing the matched predicted labels.

chord_length *Chord Length Statistic*

Description

Calculates an observation-level distance measure using chord lengths between angular values on the unit circle.

Usage

```
chord_length(theta)
```

Arguments

theta	Numeric vector containing angles in radians.

Value

Maximum normalized chord-length value.

circ_mean *Circular Mean*

Description

Calculates the weighted or unweighted mean direction of angular values.

Usage

```
circ_mean(theta, weights = NULL)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Optional numeric vector containing observation weights.

Value

Mean direction in radians between -pi and pi.

circ_r *Mean Resultant Length*

Description

Measures the concentration of angular values around their mean direction. Values close to 1 indicate strong concentration, while values close to 0 indicate greater angular dispersion.

Usage

```
circ_r(theta, weights = NULL)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Optional numeric vector containing observation weights.

Value

Numeric value between 0 and 1.

circ_std *Circular Standard Deviation*

Description

Calculates circular standard deviation from the mean resultant length. Larger values indicate greater angular dispersion.

Usage

```
circ_std(theta, weights = NULL)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Optional numeric vector containing observation weights.

Value

Non-negative numeric value.

circ_var *Circular Variance*

Description

Measures angular dispersion using the mean resultant length. Values close to 0 indicate low dispersion, while values close to 1 indicate high dispersion.

Usage

```
circ_var(theta, weights = NULL)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Optional numeric vector containing observation weights.

Value

Numeric value between 0 and 1.

circular_distance_measures *Calculate Circular Distance Measures*

Description

Calculates one or more circular statistics for the same angular vector.

Usage

```
circular_distance_measures(
  theta,
  weights = NULL,
 measures = c("mardia_kurtosis", "M_statistic", "a_statistic", "a_star_statistic",
    "chord_length")
)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Optional numeric vector containing observation weights. Weights are required for Mardia kurtosis and the M statistic.
measures	Character vector containing the names of the measures to calculate.

Value

Named numeric vector containing the selected measure values.

---

cluster_accuracy *Clustering Accuracy*

Description

Calculates clustering accuracy after matching predicted cluster labels to the known class labels.

Usage

```
cluster_accuracy(true_labels, pred_labels)
```

Arguments

true_labels	Vector containing the known class labels.
pred_labels	Vector containing the predicted cluster labels.

Value

A numeric accuracy value between 0 and 1.

---

compare_methods *Compare Feature and Clustering Methods*

Description

Applies multiple feature extraction and clustering methods to the same dataset. Each feature and clustering combination is evaluated using the selected performance measures.

Usage

```
compare_methods(
  data,
  feature_methods,
  cluster_methods,
  metrics = c("dbi", "n_clusters", "n_noise"),
  true_labels = NULL,
  normalize = NULL,
  verbose = TRUE
)
```

Arguments

`data`	Numeric matrix or data frame containing the input data.
`feature_methods`	Named list of feature extraction functions. Each function must accept data as input and return a feature matrix.
`cluster_methods`	Named list of clustering method specifications. Each entry must contain a clustering function and may contain its parameter values.
`metrics`	Character vector containing the evaluation measures. Available options are "dbi", "accuracy", "n_clusters", and "n_noise".
`true_labels`	Optional vector containing the known class labels. It is required when accuracy is selected.
`normalize`	Optional normalization method passed to prepare_features. Available options are "zscore", "range", or NULL.
`verbose`	Logical value indicating whether progress information should be displayed.

Value

A data frame containing one row for each feature and clustering method combination and the selected evaluation results.

`compute_phase` *Instantaneous Phase*

Description

Calculates instantaneous phase from the Hilbert analytic signal.

Usage

```
compute_phase(X, axis = c("time", "feature"))
```

Arguments

`X`	Numeric vector, matrix, or data frame.
`axis`	Direction in which the Hilbert transform is applied. Available options are "time" and "feature".

Value

A numeric matrix with the same dimensions as the input. Phase values are returned in radians between -pi and pi.

davies_bouldin *Davies-Bouldin Index*

Description

Calculates the Davies-Bouldin Index for a clustering result. The index compares within-cluster spread with the distance between cluster centers. Lower values indicate more compact and separated clusters.

Usage

```
davies_bouldin(X, labels, noise_label = 0)
```

Arguments

X	Numeric feature matrix. Rows represent observations and columns represent features.
labels	Integer vector containing the cluster label of each observation.
noise_label	Label used for noise observations. Default is 0.

Value

A numeric value containing the Davies-Bouldin Index. Returns NA when fewer than two clusters remain after removing noise.

extract_circular_features *Extract Circular Features*

Description

Extracts row-based circular features from a phase matrix. Each row is treated as one observation and each column as one phase value.

Usage

```
extract_circular_features(phase)
```

Arguments

phase	Numeric matrix containing phase values in radians.

Value

Numeric matrix containing mean phase, chord-based phase difference, and circular correlation distance for each row.

find_elbow *Find an Elbow Point*

Description

Finds the point with the largest perpendicular distance from the line connecting the first and last values of a sorted sequence.

Usage

```
find_elbow(values)
```

Arguments

values	Numeric vector sorted in ascending order.

Value

Integer index of the detected elbow point.

first_difference *First Differences*

Description

Calculates the difference between consecutive observations in each column. A row of zeros is added at the beginning so that the output has the same number of rows as the input.

Usage

```
first_difference(X)
```

Arguments

X	Numeric vector, matrix, or data frame.

Value

A numeric matrix containing first differences.

flatten_with_zones *Flatten Data with Group Labels*

Description

Converts a wide matrix into a single value vector and creates a matching group label for each original column.

Usage

```
flatten_with_zones(X)
```

Arguments

X	Numeric matrix. Rows represent observations and columns represent groups or zones.

Value

A list containing the flattened values and their corresponding group labels.

label_by_quantile *Create Labels from Quantile Groups*

Description

Divides numeric values into groups using selected quantile cut points and assigns an integer label to each value.

Usage

```
label_by_quantile(values, probs = c(1/3, 2/3))
```

Arguments

values	Numeric vector to be labelled.
probs	Numeric vector containing quantile cut points between 0 and 1. The default values create three groups.

Value

An integer vector containing one group label for each value.

M_statistic *M Statistic*

Description

Calculates a leave-one-out circular statistic that measures how strongly one angular observation affects the mean resultant length.

Usage

```
M_statistic(theta, weights)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Numeric vector containing observation weights.

Value

Maximum M statistic across all observations.

mardia_kurtosis *Mardia Kurtosis*

Description

Calculates a weighted multivariate kurtosis measure after representing angular values as points on the unit circle.

Usage

```
mardia_kurtosis(theta, weights)
```

Arguments

theta	Numeric vector containing angles in radians.
weights	Numeric vector containing observation weights.

Value

Numeric kurtosis value.

normalize_features *Normalize Feature Columns*

Description

Normalizes each feature column using z-score or range scaling.

Usage

```
normalize_features(X, method = c("zscore", "range"))
```

Arguments

X	Numeric matrix or data frame. Rows represent observations and columns represent features.
method	Normalization method. Available options are "zscore" and "range".

Value

Numeric matrix with the same dimensions as the input.

plot_clusters_pca *Plot Clusters Using PCA*

Description

Projects a feature matrix onto its first two principal components and displays the clustering result in two dimensions.

Usage

```
plot_clusters_pca(X, labels, noise_label = 0)
```

Arguments

X	Numeric feature matrix. Rows represent observations and columns represent features.
labels	Cluster label assigned to each observation.
noise_label	Label used for noise observations. Default is 0.

Value

A ggplot object containing the two-dimensional cluster graph.

plot_k_distance *Plot k-Distance Graph*

Description

Creates a sorted k-nearest-neighbor distance graph to support the selection of the epsilon parameter for density-based clustering.

Usage

```
plot_k_distance(X, k = 5, distance = c("euclidean", "manhattan"))
```

Arguments

X	Numeric feature matrix. Rows represent observations and columns represent features.
k	Number of nearest neighbors used in the distance calculation.
distance	Distance measure. Available options are "euclidean" and "manhattan".

Value

A ggplot object containing the sorted k-distance graph.

plot_reachability *Plot OPTICS Reachability*

Description

Creates a reachability graph from an OPTICS result. Lower values generally represent denser regions, while higher values may indicate sparse regions or transitions between clusters.

Usage

```
plot_reachability(optics_result, log_scale = FALSE, epsilon = NULL)
```

Arguments

optics_result	Result returned by run_optics.
log_scale	Logical value indicating whether reachability values should be shown on a logarithmic scale.
epsilon	Optional numeric value displayed as a horizontal reference line.

Value

A ggplot object containing the reachability graph.

power_consumption *Tetouan Power Consumption Data*

Description

Electricity consumption and weather measurements recorded at 10-minute intervals in three zones of Tetouan, Morocco.

Usage

```
power_consumption
```

Format

A data frame with 13,906 rows and 9 variables:

Datetime Date and time of the observation.

Temperature Ambient temperature.

Humidity Relative humidity.

WindSpeed Wind speed.

GeneralDiffuseFlows General diffuse solar radiation.

DiffuseFlows Diffuse solar radiation.

PowerConsumption_Zone1 Electricity consumption in Zone 1.

PowerConsumption_Zone2 Electricity consumption in Zone 2.

PowerConsumption_Zone3 Electricity consumption in Zone 3.

Source

UCI Machine Learning Repository, Tetouan City Power Consumption dataset.

prepare_features *Prepare a Feature Matrix*

Description

Converts input features to a matrix, replaces missing values, and optionally applies normalization before clustering.

Usage

```
prepare_features(X, normalize = NULL)
```

Arguments

X	Numeric matrix or data frame containing extracted features.
normalize	Optional normalization method. Available options are "zscore", "range", or NULL.

Value

A numeric matrix prepared for further analysis.

---

rolling_stats *Rolling Statistics*

Description

Calculates moving summary statistics for each column of a numeric signal. The window is centered on each observation and becomes shorter near the beginning and end of the signal.

Usage

```
rolling_stats(X, window_size = 10, stats = c("mean", "sd", "max", "min"))
```

Arguments

X	Numeric vector, matrix, or data frame. Rows represent observations and columns represent signals.
window_size	Positive integer defining the moving window length.
stats	Character vector containing the statistics to calculate. Available options are "mean", "sd", "max", and "min".

Value

A named list containing one numeric matrix for each selected statistic.

---

run_dbscan *DBSCAN Clustering*

Description

Applies DBSCAN to a numeric feature matrix. The algorithm identifies dense groups of observations and labels observations outside these groups as noise.

Usage

```
run_dbscan(X, eps, min_pts)
```

Arguments

X	Numeric matrix or data frame. Rows represent observations and columns represent features.
eps	Maximum distance used to define the neighbourhood of a point.
min_pts	Minimum number of points required to form a dense region.

Value

A list containing cluster labels, the number of clusters, the number of noise observations, the method name, and parameter values.

Examples

```
set.seed(1)

X <- rbind(
  matrix(rnorm(60, mean = 0, sd = 0.5), ncol = 2),
  matrix(rnorm(60, mean = 5, sd = 0.5), ncol = 2)
)

result <- run_dbscan(X, eps = 1, min_pts = 5)

result$n_clusters
result$n_noise
table(result$cluster)
```

---

run_optics *OPTICS Clustering*

Description

Applies an OPTICS-based procedure to a numeric feature matrix. The function orders observations according to local density and calculates reachability values used to represent cluster structure.

Usage

```
run_optics(X, eps, min_pts, distance = c("euclidean", "manhattan"))
```

Arguments

X	Numeric matrix or data frame. Rows represent observations and columns represent features.
eps	Maximum distance used to find neighbouring observations.
min_pts	Minimum number of neighbouring points required to define a dense region.
distance	Distance measure. Available options are "euclidean" and "manhattan".

Value

A list containing cluster labels, the number of clusters, the number of noise observations, reachability values, observation order, the method name, and parameter values.

Examples

```
set.seed(1)

X <- rbind(
  matrix(rnorm(60, mean = 0, sd = 0.5), ncol = 2),
  matrix(rnorm(60, mean = 5, sd = 0.5), ncol = 2)
)

result <- run_optics(
  X,
  eps = 1.5,
  min_pts = 5,
  distance = "euclidean"
)

result$n_clusters
result$n_noise
head(result$reachability)
```

---

segment_signal *Segment a Signal*

Description

Divides a numeric signal into consecutive non-overlapping windows. Values that do not complete a full window are omitted.

Usage

```
segment_signal(x, window_size)
```

Arguments

x	Numeric vector containing the signal.
window_size	Positive integer defining the number of values in each window.

Value

A list of numeric vectors with equal window lengths.

`select_numeric_columns` *Select Numeric Columns*

Description

Extracts numeric columns from a data frame and returns them as a matrix.

Usage

```
select_numeric_columns(data)
```

Arguments

`data`	Data frame containing numeric and non-numeric columns.

Value

A numeric matrix containing only the numeric columns.

`steel_industry` *Steel Industry Energy Consumption Data*

Description

Energy consumption and production-related measurements recorded at 15-minute intervals during 2018 at a steel production facility in South Korea.

Usage

```
steel_industry
```

Format

A data frame with 35,040 rows and 11 variables:

date Date and time of the observation.
Usage_kWh Electricity consumption in kilowatt-hours.
Lagging_Current_Reactive.Power_kVarh Lagging reactive power.
Leading_Current_Reactive_Power_kVarh Leading reactive power.
CO2.tCO2. Carbon dioxide emissions.
Lagging_Current_Power_Factor Lagging current power factor.
Leading_Current_Power_Factor Leading current power factor.
NSM Number of seconds since midnight.
WeekStatus Weekday or weekend category.
Day_of_week Day of the week.
Load_Type Electricity load category.

Source

UCI Machine Learning Repository, Steel Industry Energy Consumption dataset.

`thin_data` *Thin a Dataset*

Description

Reduces the number of rows by keeping every selected row interval.

Usage

```
thin_data(data, step = 1)
```

Arguments

`data`	Data frame or matrix.
`step`	Positive integer indicating the interval between retained rows. A value of 1 keeps all rows.

Value

A data frame or matrix containing the selected rows.

`wavelet_approx` *Multi-Level Wavelet Approximation*

Description

Applies a multi-level discrete wavelet decomposition and reconstructs the signal using only the approximation information.

Usage

```
wavelet_approx(x, wavelet = "d4", n_levels = 2)
```

Arguments

`x`	Numeric vector containing the signal.
`wavelet`	Wavelet filter name accepted by the waveslim package.
`n_levels`	Positive integer defining the number of decomposition levels.

Value

Numeric vector containing the reconstructed approximation signal.

wavelet_transform *Single-Level Wavelet Transform*

Description

Applies a single-level discrete wavelet transform to a numeric signal. The function returns approximation and detail coefficients.

Usage

```
wavelet_transform(x, wavelet = c("db1", "db4"))
```

Arguments

x	Numeric vector containing the signal.
wavelet	Wavelet type. Available options are "db1" and "db4".

Value

A list containing the approximation coefficients in cA and the detail coefficients in cD.

window_moments *Window Summary Statistics*

Description

Divides a numeric signal into consecutive non-overlapping windows and calculates selected summary statistics for each window.

Usage

```
window_moments(
  x,
  window_size = 4,
  stats = c("mean", "sd", "skewness", "kurtosis")
)
```

Arguments

x	Numeric vector containing the signal.
window_size	Positive integer defining the number of observations in each window.
stats	Character vector containing the statistics to calculate. Available options are "mean", "sd", "skewness", "kurtosis", "range", and "energy".

Value

A numeric matrix in which rows represent windows and columns represent the selected statistics.

wrap_to_pi *Wrap Angles*

Description

Converts angular values to the standard interval from -pi to pi.

Usage

```
wrap_to_pi(x)
```

Arguments

x	Numeric vector or matrix containing angles in radians.

Value

Values with the same dimensions as the input, wrapped to the interval from -pi inclusive to pi exclusive.

Index