Package 'RegimeChange'

May 7, 2026

Title Comprehensive Regime Change Detection in Time Series

Version 0.1.1

Description A unified framework for detecting regime changes (changepoints) in time series data. Implements both frequentist methods including Cumulative Sum (CUSUM, Page (1954) <doi:10.1093/biomet/41.1-2.100>), Pruned Exact Linear Time (PELT, Killick, Fearnhead, and Eckley (2012) <doi:10.1080/01621459.2012.737745>), Binary Segmentation, and Wild Binary Segmentation, as well as Bayesian methods such as Bayesian Online Changepoint Detection (BOCPD, Adams and MacKay (2007) <doi:10.48550/arXiv.0710.3742> and Shiryaev-Roberts. Supports offline analysis for retrospective detection and online monitoring for real-time surveillance. Provides rigorous uncertainty quantification through confidence intervals and posterior distributions. Handles univariate and multivariate series with detection of changes in mean, variance, trend, and distributional properties.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, ggplot2 (>= 3.4.0), rlang (>= 1.1.0), cli (>= 3.6.0), magrittr

Suggests testthat (>= 3.0.0), knitr, rmarkdown, plotly, patchwork, covr, JuliaCall (>= 0.17.0), keras (>= 2.9.0), tensorflow (>= 2.9.0), reticulate (>= 1.26)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/IsadoreNabi/RegimeChange

BugReports https://github.com/IsadoreNabi/RegimeChange/issues

LazyData true

SystemRequirements Julia (>= 1.6) for optional high-performance backend, Python (>= 3.8) with TensorFlow for deep learning methods

NeedsCompilation no

Author José Mauricio Gómez Julián [aut, cre] (ORCID:
<https://orcid.org/0009-0000-2412-3150>)

Maintainer José Mauricio Gómez Julián <isadorenabi@pm.me>

Repository CRAN

Date/Publication 2026-02-13 16:34:59 UTC

Contents

`RegimeChange-package` *RegimeChange: Comprehensive Regime Change Detection in Time Series*

Description

A unified framework for detecting regime changes (changepoints) in time series data. Implements both frequentist methods including Cumulative Sum (CUSUM, Page (1954) doi:10.1093/biomet/41.12.100), Pruned Exact Linear Time (PELT, Killick, Fearnhead, and Eckley (2012) doi:10.1080/01621459.2012.737745), Binary Segmentation, and Wild Binary Segmentation, as well as Bayesian methods such as Bayesian Online Changepoint Detection (BOCPD, Adams and MacKay (2007) https://arxiv.org/abs/0710.3742) and Shiryaev-Roberts. Supports offline analysis for retrospective detection and online monitoring for real-time surveillance. Provides rigorous uncertainty quantification through confidence intervals and posterior distributions. Handles univariate and multivariate series with detection of changes in mean, variance, trend, and distributional properties.

Author(s)

Maintainer: José Mauricio Gómez Julián <isadorenabi@pm.me> (ORCID)

See Also

Useful links:

- https://github.com/IsadoreNabi/RegimeChange
- Report bugs at https://github.com/IsadoreNabi/RegimeChange/issues

adjusted_rand_index *Adjusted Rand Index*

Description

Rand Index corrected for chance agreement.

Usage

```
adjusted_rand_index(detected, true_cp, n)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
n	Total number of observations

Value

Adjusted Rand Index (can be negative, 1 is perfect)

autoencoder_detect *Autoencoder-based Changepoint Detection*

Description

Detects changepoints by identifying regions where reconstruction error is anomalously high, indicating the model (trained on normal patterns) fails to reconstruct the data.

Usage

```
autoencoder_detect(
  data,
  window_size = 50,
  latent_dim = 10,
  hidden_dims = c(32, 16),
  epochs = 100,
  batch_size = 32,
  threshold = NULL,
  contamination = 0.1,
  variational = FALSE,
  verbose = FALSE
)
```

Arguments

data	Numeric vector of time series data
window_size	Size of sliding window (default: 50)
latent_dim	Dimension of latent space (default: 10)
hidden_dims	Hidden layer dimensions (default: c(32, 16))
epochs	Training epochs (default: 100)
batch_size	Batch size (default: 32)
threshold	Threshold for anomaly detection. NULL for automatic selection using 3-sigma rule.
contamination	Expected proportion of anomalies for threshold selection (default: 0.1)
variational	Use Variational Autoencoder (default: FALSE)
verbose	Show training progress (default: FALSE)

Value

List with:

changepoints	Detected changepoint locations
reconstruction_error	Per-window reconstruction error
threshold	Threshold used for detection
model	Trained Keras model

Examples

```
## Not run:
data <- c(rnorm(100), rnorm(100, mean = 3), rnorm(100))
result <- autoencoder_detect(data, window_size = 30)
plot(result$reconstruction_error, type = "l")
abline(v = result$changepoints, col = "red")

## End(Not run)
```

---

benchmark_backends *Benchmark Julia vs R Performance*

Description

Compare execution time between Julia and R implementations.

Usage

```
benchmark_backends(data, method = "pelt", n_reps = 10, ...)
```

Arguments

data	Test data.
method	Method to benchmark.
n_reps	Number of repetitions.
...	Additional arguments for the method.

Value

Data frame with timing results (invisibly).

Examples

```
data <- c(rnorm(500), rnorm(500, 3))
benchmark_backends(data, method = "pelt", penalty = log(1000))
```

---

binary_segmentation *Binary Segmentation Changepoint Detection*

Description

Recursive binary segmentation for changepoint detection. Greedy algorithm that recursively splits at the best changepoint.

Usage

```
binary_segmentation(
  data,
  type = "both",
  penalty = "BIC",
  min_segment = 2,
  n_changepoints = "multiple",
  threshold = NULL,
  ...
)
```

Arguments

data	Numeric vector or matrix
type	Type of change to detect
penalty	Penalty for adding changepoints
min_segment	Minimum segment length
n_changepoints	Maximum number of changepoints to detect
threshold	Significance threshold
...	Additional arguments

Value

List with changepoints

Examples

```
data <- c(rnorm(100), rnorm(100, mean = 2))
result <- binary_segmentation(data)
```

---

bocpd *Bayesian Online Changepoint Detection*

Description

Implements the BOCPD algorithm of Adams and MacKay (2007). Maintains a posterior distribution over the run length (time since last changepoint) and updates it online as new observations arrive.

Usage

```
bocpd(
  data,
  type = "both",
  prior = NULL,
  hazard = NULL,
  threshold = 0.3,
  truncate_run_length = NULL,
  ...
)
```

Arguments

data	Numeric vector or matrix
type	Type of change to detect (currently supports "both")
prior	Prior specification (from normal_gamma(), normal_known_var(), etc.)
hazard	Hazard prior (from geometric_hazard(), etc.) or numeric hazard rate
threshold	Probability threshold for declaring a changepoint
truncate_run_length	Maximum run length to track (for efficiency)
...	Additional arguments

Value

List containing:

- changepoints: Detected changepoint locations
- posterior: Matrix of run length posteriors over time
- prob_change: Probability of changepoint at each time
- map_run_length: Maximum a posteriori run length at each time

References

Adams, R. P. and MacKay, D. J. C. (2007). Bayesian Online Changepoint Detection. arXiv:0710.3742

Examples

```
data <- c(rnorm(100, mean = 0), rnorm(100, mean = 3))

result <- bocpd(data)

result <- bocpd(data, prior = normal_gamma(mu0 = 0, kappa0 = 1,
                                           alpha0 = 1, beta0 = 1))


result <- bocpd(data, hazard = 0.01)
```

`compare_methods` *Compare Multiple Detection Methods*

Description

Runs multiple detection methods on the same data and compares results.

Usage

```
compare_methods(
  data,
  methods = c("pelt", "binseg", "wbs", "bocpd"),
  true_changepoints = NULL,
  tolerance = 5,
  ...
)
```

Arguments

`data`	Numeric vector or matrix
`methods`	Character vector of method names
`true_changepoints`	Vector of true changepoints (for evaluation)
`tolerance`	Tolerance for evaluation metrics
`...`	Additional arguments passed to detect_regimes

Value

Object of class "regime_comparison"

Examples

```
true_cp <- c(50, 100)
data <- c(rnorm(50), rnorm(50, mean = 2), rnorm(50))

comparison <- compare_methods(
  data,
  methods = c("pelt", "binseg"),
  true_changepoints = true_cp
)
```

`constant_hazard` *Constant Hazard Prior (Alias for Geometric)*

Description

Constant Hazard Prior (Alias for Geometric)

Usage

```
constant_hazard(lambda = 0.01)
```

Arguments

`lambda`	Hazard rate (probability of changepoint at each step). Should be between 0 and 1. Smaller values expect fewer changepoints.

Value

An object of class "hazard_prior"

covering_metric *Covering Metric*

Description

Measures how well detected segments cover true segments using IoU.

Usage

```
covering_metric(detected, true_cp, n)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
n	Total number of observations

Value

Covering metric (0 to 1)

cpc_detect *Contrastive Predictive Coding for Changepoint Detection*

Description

Uses self-supervised contrastive learning to detect changepoints by identifying where the learned representations change significantly.

Usage

```
cpc_detect(
  data,
  window_size = 64,
  encoding_dim = 32,
  n_negative = 10,
  prediction_steps = 5,
  epochs = 100,
  threshold = NULL,
  verbose = FALSE
)
```

Arguments

data	Numeric vector
window_size	Window size (default: 64)
encoding_dim	Encoding dimension (default: 32)
n_negative	Number of negative samples (default: 10)
prediction_steps	Future steps to predict (default: 5)
epochs	Training epochs (default: 100)
threshold	Detection threshold for representation distance
verbose	Show progress

Value

List with changepoints and learned encodings

References

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding

---

crops_detect *CROPS - Changepoints for a Range of Penalties*

Description

Efficiently computes optimal segmentations for a range of penalty values, useful for model selection.

Usage

```
crops_detect(
  data,
  penalty_range = c(0.1, 100),
  method = "pelt",
  min_segment = 2
)
```

Arguments

data	Numeric vector
penalty_range	Vector of length 2: c(min_penalty, max_penalty)
method	Segmentation method ("pelt" or "fpop")
min_segment	Minimum segment length

Value

List with:

penalties	Penalty values tested
n_changepoints	Number of changepoints for each penalty
changepoints	List of changepoint vectors
costs	Optimal costs

References

Haynes, K., Eckley, I. A., and Fearnhead, P. (2017). Computationally efficient changepoint detection for a range of penalties. Journal of Computational and Graphical Statistics, 26(1), 134-143.

---

cusum *CUSUM Changepoint Detection*

Description

Detects changepoints using the Cumulative Sum (CUSUM) statistic. Foundational method for detecting changes in mean.

Usage

```
cusum(
  data,
  type = "mean",
  threshold = 4,
  mode = "offline",
  mu0 = NULL,
  sigma = NULL,
  ...
)
```

Arguments

data	Numeric vector
type	Type of change ("mean", "variance", "both")
threshold	Detection threshold (alarm when statistic exceeds this)
mode	"offline" for retrospective or "online" for sequential
mu0	Target mean under null hypothesis (for online mode)
sigma	Known standard deviation (if NULL, estimated from data)
...	Additional arguments

Value

List with changepoints and statistics

Examples

```
data <- c(rnorm(50), rnorm(50, mean = 2))
result <- cusum(data)
```

`detect_pelt` *Detect changepoints using enhanced PELT algorithm*

Description

Pruned Exact Linear Time (PELT) algorithm for optimal changepoint detection with enhancements for autocorrelated data and numerical stability.

Usage

```
detect_pelt(
  data,
  type = "both",
  penalty = "MBIC",
  min_segment = 2,
  robust = FALSE,
  correct_ar = FALSE,
  merge_close = NULL,
  ...
)
```

Arguments

`data`	Numeric vector or matrix. For matrices, rows are observations.
`type`	Type of change to detect: `"mean"`, `"var"`, or `"both"`.
`penalty`	Penalty type (`"MBIC"`, `"BIC"`, `"AIC"`) or numeric value.
`min_segment`	Minimum segment length (default: 2).
`robust`	Logical or character. FALSE for standard, TRUE for moderate robustness, or one of "mild", "moderate", "aggressive", "auto" for specific levels.
`correct_ar`	Logical. Apply pre-whitening for autocorrelated data. Default: FALSE.
`merge_close`	Integer or NULL. Merge changepoints within this distance. Default: NULL.
`...`	Additional arguments (currently unused).

Details

This implementation includes enhancements over standard PELT:

Pre-whitening for autocorrelated data: When `correct_ar = TRUE`, the function estimates the AR(1) coefficient and transforms the data via $y'_t = y_t - \rho y_{t-1}$ before applying PELT.

Configurable robustness: The `robust` parameter accepts:

- `FALSE`: Standard MLE-based estimation
- `TRUE` or `"moderate"`: 10\
- `"mild"`: 5\
- `"aggressive"`: 15\
- `"auto"`: Automatically select based on data characteristics

Value

A list with components:

changepoints Integer vector of detected changepoint locations

n_changepoints Number of changepoints detected

cost Final cost value

ar_coefficient Estimated AR(1) coefficient (if `correct_ar = TRUE`)

prewhitened Logical indicating if pre-whitening was applied

robust Logical indicating if robust estimation was used

robust_level Character indicating robustness level used

information_criterion Reserved for future use

References

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. JASA, 107(500), 1590-1598.

Examples

```
data <- c(rnorm(100, 0), rnorm(100, 3))
result <- detect_pelt(data, type = "mean")


data_contaminated <- c(rnorm(100, 0), rnorm(100, 3))
data_contaminated[sample(200, 10)] <- rnorm(10, 0, 10)
result_robust <- detect_pelt(data_contaminated, robust = TRUE)


result_auto <- detect_pelt(data_contaminated, robust = "auto")
```

`detect_regimes` *Detect Regime Changes in Time Series*

Description

Main function for detecting regime changes (changepoints) in time series data. Supports multiple detection methods, both frequentist and Bayesian, and can operate in offline (retrospective) or online (sequential) modes.

Usage

```
detect_regimes(
  data,
  method = c("pelt", "bocpd", "cusum", "binseg", "wbs", "shiryaev", "ensemble"),
  type = c("both", "mean", "variance", "trend", "distribution"),
  mode = c("offline", "online"),
  n_changepoints = "multiple",
  penalty = "BIC",
  min_segment = 2,
  prior = NULL,
  hazard = NULL,
  threshold = NULL,
  uncertainty = TRUE,
  bootstrap_reps = 200,
  ...
)
```

Arguments

`data`	Numeric vector, time series (ts object), or matrix for multivariate data. For matrices, rows are observations and columns are variables.
`method`	Detection method. One of: • `"pelt"`: Pruned Exact Linear Time (default for offline) • `"bocpd"`: Bayesian Online Changepoint Detection • `"cusum"`: Cumulative Sum • `"binseg"`: Binary Segmentation • `"wbs"`: Wild Binary Segmentation • `"shiryaev"`: Shiryaev-Roberts procedure • `"ensemble"`: Combination of multiple methods
`type`	Type of change to detect: • `"mean"`: Changes in mean only • `"variance"`: Changes in variance only • `"both"`: Changes in mean and/or variance (default) • `"trend"`: Changes in linear trend

- `"distribution"`: Non-parametric distributional changes

`mode` Operation mode:

- `"offline"`: Retrospective analysis with full data (default)
- `"online"`: Sequential analysis for monitoring

`n_changepoints` Expected number of changepoints:

- `"single"`: Detect at most one changepoint
- `"multiple"`: Detect multiple changepoints (default)
- An integer: Detect exactly this many changepoints

`penalty` Penalty for model complexity (offline methods):

- `"BIC"`: Bayesian Information Criterion (default)
- `"AIC"`: Akaike Information Criterion
- `"MBIC"`: Modified BIC
- `"MDL"`: Minimum Description Length
- A number: Manual penalty value

`min_segment` Minimum segment length (number of observations)

`prior` Prior specification for Bayesian methods (from prior functions)

`hazard` Hazard prior for changepoint occurrence (Bayesian methods)

`threshold` Detection threshold (for online/CUSUM methods)

`uncertainty` Logical; if TRUE, compute confidence intervals

`bootstrap_reps` Number of bootstrap replicates for uncertainty (if uncertainty = TRUE)

`...` Additional arguments passed to specific methods

Value

An object of class `"regime_result"` containing:

- `changepoints`: Vector of detected changepoint locations
- `n_changepoints`: Number of changepoints detected
- `segments`: List of segment information (start, end, parameters)
- `confidence_intervals`: Confidence intervals for changepoint locations
- `existence_probability`: Probability that each changepoint exists
- `posterior`: Posterior distribution (Bayesian methods)
- `information_criterion`: BIC/AIC values
- `method`: Method used
- `call`: The function call

Examples

```
set.seed(123)
data <- c(rnorm(100, mean = 0), rnorm(100, mean = 3))


result <- detect_regimes(data)
print(result)
plot(result)


result <- detect_regimes(data, method = "bocpd",
                         prior = normal_gamma())


result <- detect_regimes(data, method = "cusum",
                         mode = "online", threshold = 5)
```

---

economic_cycles *Economic Cycles Dataset*

Description

A simulated time series representing an economic indicator with regime changes corresponding to expansion, recession, recovery, and stable growth periods.

Usage

```
economic_cycles
```

Format

A time series object of length 500 with attributes:

true_changepoints Vector of true changepoint locations: c(120, 250, 380)

description Description of the dataset

regimes Names of the regimes: Expansion, Recession, Recovery, Stable Growth

Details

The data simulates monthly economic data from 2000-2041 with four distinct regimes:

- Regime 1 (1-120): Expansion - positive trend, low volatility
- Regime 2 (121-250): Recession - negative trend, high volatility
- Regime 3 (251-380): Recovery - positive trend, medium volatility
- Regime 4 (381-500): Stable Growth - low trend, low volatility

Source

Simulated data for package examples

Examples

```
data(economic_cycles)
result <- detect_regimes(economic_cycles, method = "pelt")
plot(result)

# Compare with true changepoints
true_cps <- attr(economic_cycles, "true_changepoints")
evaluate(result, true_changepoints = true_cps)
```

---

edivisive_detect *E-Divisive Changepoint Detection*

Description

Nonparametric changepoint detection using energy statistics. Can detect changes in any aspect of the distribution.

Usage

```
edivisive_detect(data, min_segment = 5, alpha = 0.05, n_perm = 199)
```

Arguments

`data`	Numeric vector or matrix
`min_segment`	Minimum segment length
`alpha`	Significance level for permutation tests
`n_perm`	Number of permutations

Value

List with changepoints and test statistics

References

Matteson, D. S., and James, N. A. (2014). A nonparametric approach for multiple change point analysis of multivariate data. Journal of the American Statistical Association, 109(505), 334-345.

ensemble_dl_detect *Ensemble Deep Learning Detection*

Description

Combines multiple deep learning methods for robust detection.

Usage

```
ensemble_dl_detect(
  data,
  methods = c("autoencoder", "tcn", "transformer"),
  min_agreement = 2,
  ...
)
```

Arguments

data	Numeric vector
methods	Vector of methods to use (default: all)
min_agreement	Minimum number of methods that must agree
...	Additional arguments passed to individual methods

Value

List with consensus changepoints and individual results

evaluate *Evaluate Changepoint Detection Results*

Description

Comprehensive evaluation of detection results against known ground truth. Computes multiple metrics for localization, segmentation, and detection.

Usage

```
evaluate(result, true_changepoints, n = NULL, tolerance = 5)
```

Arguments

result	A regime_result object or vector of changepoint locations
true_changepoints	Vector of true changepoint locations
n	Total number of observations (required if result is a vector)
tolerance	Tolerance window for matching changepoints (for F1 score)

Value

A list of class "regime_evaluation" containing all metrics

Examples

```
true_cp <- c(50, 100)
data <- c(rnorm(50), rnorm(50, mean = 2), rnorm(50))

result <- detect_regimes(data)
evaluation <- evaluate(result, true_cp)
print(evaluation)
```

---

evaluation *Evaluation Metrics for Changepoint Detection*

Description

Functions for evaluating changepoint detection results against ground truth. Includes metrics for localization accuracy, segmentation quality, and detection performance.

---

f1_score *F1 Score with Tolerance*

Description

Harmonic mean of precision and recall.

Usage

```
f1_score(detected, true_cp, tolerance = 5)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
tolerance	Maximum distance for a match

Value

F1 score (0 to 1)

Examples

```
f1_score(c(48, 102, 150), c(50, 100), tolerance = 5)
```

`fpop_detect` *FPOP - Functional Pruning Optimal Partitioning*

Description

More efficient than PELT for certain data types by maintaining piecewise quadratic cost functions instead of just minimum values. Achieves O(n) complexity in practice.

Usage

```
fpop_detect(data, penalty = "bic", min_segment = 2, cost_type = "mean")
```

Arguments

`data`	Numeric vector of observations
`penalty`	Penalty for adding a changepoint. Can be: • `"bic"`: log(n) • `"aic"`: 2 • `"mbic"`: 3*log(n) • numeric: Custom penalty value
`min_segment`	Minimum segment length (default: 2)
`cost_type`	Type of cost function: • `"mean"`: Gaussian mean change (default) • `"meanvar"`: Gaussian mean and variance change • `"poisson"`: Poisson rate change

Value

List with:

`changepoints`	Vector of changepoint locations
`cost`	Final optimal cost
`n_candidates`	Average candidates per time point (efficiency metric)

References

Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2017). On optimal multiple changepoint algorithms for large data. Statistics and Computing, 27(2), 519-533.

Examples

```
data <- c(rnorm(100, 0), rnorm(100, 3), rnorm(100, 1))
result <- fpop_detect(data, penalty = "bic")
print(result$changepoints)
```

geometric_hazard *Geometric Hazard Prior*

Description

Creates a geometric (constant hazard) prior for the changepoint process. This implies that the probability of a changepoint at each time step is constant and independent.

Usage

```
geometric_hazard(lambda = 0.01)
```

Arguments

lambda	Hazard rate (probability of changepoint at each step). Should be between 0 and 1. Smaller values expect fewer changepoints.

Details

Under a geometric hazard, the expected run length (time between changepoints) is 1/lambda. For example, lambda = 0.01 expects a changepoint every 100 observations on average.

Value

An object of class "hazard_prior"

Examples

```
hazard <- geometric_hazard(lambda = 0.01)

hazard <- geometric_hazard(lambda = 0.1)
```

hausdorff_distance *Hausdorff Distance Between Changepoint Sets*

Description

Computes the Hausdorff distance between detected and true changepoints. The Hausdorff distance is the maximum distance from a point in one set to the nearest point in the other set.

Usage

```
hausdorff_distance(detected, true_cp)
```

Arguments

`detected`	Vector of detected changepoint locations
`true_cp`	Vector of true changepoint locations

Value

Hausdorff distance (non-negative number)

Examples

```
hausdorff_distance(c(48, 102), c(50, 100))
```

`industrial_sensor` *Industrial Sensor Dataset*

Description

A simulated time series from a manufacturing process with abrupt changes representing process shifts and equipment states.

Usage

```
industrial_sensor
```

Format

A time series object of length 600 with attributes:

true_changepoints Vector of true changepoint locations: c(150, 300, 450)

description Description of the dataset

regimes Names of the regimes: Normal, Drift, Malfunction, Corrected

Details

The data simulates per-minute sensor measurements with four distinct regimes:

- Regime 1 (1-150): Normal operation - mean=50, sd=2
- Regime 2 (151-300): Process drift - mean=55, sd=2.5
- Regime 3 (301-450): Equipment malfunction - mean=45, sd=5
- Regime 4 (451-600): Corrected operation - mean=50, sd=1.5

The data includes autocorrelation typical of industrial processes.

Source

Simulated data for package examples

Examples

```
data(industrial_sensor)
result <- detect_regimes(industrial_sensor, method = "bocpd")
plot(result, type = "segments")
```

`init_julia` *Initialize Julia Backend*

Description

Explicitly initialize the Julia backend for improved performance. This loads the RegimeChangeJulia module and makes Julia functions available.

Usage

```
init_julia(force = FALSE, num_threads = NULL)
```

Arguments

`force`	Logical. Force reinitialization even if already initialized. Default is FALSE.
`num_threads`	Integer or NULL. Number of Julia threads to use. Default is NULL (auto-detect).

Value

Logical indicating success (invisibly).

Examples

```
## Not run:
init_julia()
julia_available()
result <- detect_regimes(data, method = "pelt")

## End(Not run)
```

inverse_gamma_var *Inverse-Gamma Prior for Variance Only*

Description

Creates an Inverse-Gamma prior for detecting changes in variance with known mean.

Usage

```
inverse_gamma_var(alpha0 = 1, beta0 = 1, known_mean = 0)
```

Arguments

alpha0	Shape parameter
beta0	Scale parameter
known_mean	Known mean of the observations

Value

An object of class "regime_prior"

julia_available *Check if Julia Backend is Available*

Description

Check if Julia Backend is Available

Usage

```
julia_available()
```

Value

Logical indicating if Julia is available and initialized.

julia_status *Get Julia Backend Status*

Description

Get Julia Backend Status

Usage

```
julia_status()
```

Value

List with Julia status information.

kernel_cpd_detect *Kernel-based Changepoint Detection*

Description

Detects changepoints using Maximum Mean Discrepancy (MMD) in a Reproducing Kernel Hilbert Space. This nonparametric approach can detect changes in any aspect of the distribution.

Usage

```
kernel_cpd_detect(
  data,
  penalty = "bic",
  kernel = "rbf",
  bandwidth = 0,
  min_segment = 5
)
```

Arguments

`data`	Numeric vector or matrix (rows = observations)
`penalty`	Penalty for changepoints (default: "bic")
`kernel`	Kernel function to use: • `"rbf"`: Radial Basis Function (Gaussian) kernel • `"linear"`: Linear kernel • `"poly"`: Polynomial kernel • `"laplacian"`: Laplacian kernel
`bandwidth`	Kernel bandwidth (0 = automatic median heuristic)
`min_segment`	Minimum segment length

Value

List with changepoints and kernel statistics

References

Arlot, S., Celisse, A., and Harchaoui, Z. (2019). A kernel multiple change-point algorithm via model selection. Journal of Machine Learning Research, 20(162), 1-56.

---

mean_absolute_error *Mean Absolute Error for Changepoints*

Description

Computes mean absolute error between matched changepoints. Unmatched changepoints are ignored.

Usage

```
mean_absolute_error(detected, true_cp)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations

Value

Mean absolute error

---

negbin_hazard *Negative Binomial Hazard Prior*

Description

Creates a negative binomial hazard prior, which allows for more flexibility in the distribution of run lengths.

Usage

```
negbin_hazard(r = 1, p = 0.01)
```

Arguments

r	Number of successes (shape parameter)
p	Probability of success

Value

An object of class "hazard_prior"

`normal_gamma` *Normal-Gamma Prior for Unknown Mean and Variance*

Description

Creates a Normal-Gamma prior specification for data with unknown mean and variance. This is conjugate for normal observations.

Usage

```
normal_gamma(mu0 = 0, kappa0 = 1, alpha0 = 1, beta0 = 1)
```

Arguments

`mu0`	Prior mean for the mean parameter
`kappa0`	Prior pseudo-observations for the mean (strength of prior)
`alpha0`	Shape parameter for the precision (inverse variance)
`beta0`	Rate parameter for the precision

Details

The Normal-Gamma prior places a joint distribution on (mu, tau) where tau = 1/sigma^2:

tau ~ Gamma(alpha0, beta0) mu | tau ~ Normal(mu0, 1/(kappa0 * tau))

The prior mean of mu is mu0, and the prior mean of sigma^2 is beta0/(alpha0-1) for alpha0 > 1.

Value

An object of class "regime_prior"

Examples

```
prior <- normal_gamma()

prior <- normal_gamma(mu0 = 0, kappa0 = 10, alpha0 = 3, beta0 = 2)
```

normal_known_var *Normal Prior for Unknown Mean with Known Variance*

Description

Creates a Normal prior specification for data with unknown mean but known variance.

Usage

```
normal_known_var(mu0 = 0, sigma0 = 1, known_var = 1)
```

Arguments

mu0	Prior mean
sigma0	Prior standard deviation for the mean
known_var	Known variance of the observations

Value

An object of class "regime_prior"

Examples

```
prior <- normal_known_var(mu0 = 0, sigma0 = 1, known_var = 1)
```

normal_wishart *Normal-Wishart Prior for Multivariate Data*

Description

Creates a Normal-Wishart prior for multivariate normal data with unknown mean vector and covariance matrix.

Usage

```
normal_wishart(mu0, kappa0 = 1, nu0 = NULL, Psi0 = NULL)
```

Arguments

mu0	Prior mean vector (d-dimensional)
kappa0	Prior pseudo-observations for the mean
nu0	Degrees of freedom for the Wishart (must be >= d)
Psi0	Scale matrix for the Wishart (d x d positive definite)

Value

An object of class "regime_prior"

Examples

```
prior <- normal_wishart(
  mu0 = c(0, 0),
  kappa0 = 1,
  nu0 = 3,
  Psi0 = diag(2)
)
```

`not_detect` *NOT - Narrowest-Over-Threshold Changepoint Detection*

Description

Detects changepoints by finding the narrowest intervals that contain significant changes, providing excellent localization.

Usage

```
not_detect(
  data,
  threshold = NULL,
  min_segment = 5,
  contrast_type = "pcwsConst"
)
```

Arguments

`data`	Numeric vector
`threshold`	Detection threshold (NULL for automatic)
`min_segment`	Minimum segment length
`contrast_type`	Type of contrast function

Value

List with changepoints and interval information

References

Baranowski, R., Chen, Y., and Fryzlewicz, P. (2019). Narrowest-over- threshold detection of multiple change points and change-point-like features. Journal of the Royal Statistical Society Series B, 81(3), 649-672.

`pelt` *Detect changepoints using PELT algorithm*

Description

Wrapper for `detect_pelt` for compatibility.

Usage

```
pelt(
  data,
  type = "both",
  penalty = "MBIC",
  min_segment = 2,
  robust = FALSE,
  correct_ar = FALSE,
  merge_close = NULL,
  ...
)
```

Arguments

`data`	Numeric vector or matrix. For matrices, rows are observations.
`type`	Type of change to detect: `"mean"`, `"var"`, or `"both"`.
`penalty`	Penalty type (`"MBIC"`, `"BIC"`, `"AIC"`) or numeric value.
`min_segment`	Minimum segment length (default: 2).
`robust`	Logical or character. FALSE for standard, TRUE for moderate robustness, or one of "mild", "moderate", "aggressive", "auto" for specific levels.
`correct_ar`	Logical. Apply pre-whitening for autocorrelated data. Default: FALSE.
`merge_close`	Integer or NULL. Merge changepoints within this distance. Default: NULL.
`...`	Additional arguments (currently unused).

Value

List with changepoints and diagnostics

See Also

`detect_pelt`

plot.regime_result *Plot Regime Change Detection Results*

Description

Create visualizations of changepoint detection results.

Usage

```
## S3 method for class 'regime_result'
plot(
  x,
  type = c("data", "segments", "posterior", "diagnostic", "runlength"),
  show_ci = TRUE,
  show_segments = TRUE,
  title = NULL,
  ...
)
```

Arguments

x	A regime_result object
type	Type of plot: • "data": Data with changepoints marked (default) • "segments": Data colored by segment • "posterior": Posterior probability of change (Bayesian methods) • "diagnostic": Diagnostic statistics • "runlength": Run length distribution (BOCPD)
show_ci	Show confidence intervals if available
show_segments	Color segments differently
title	Plot title
...	Additional arguments passed to ggplot

Value

A ggplot2 object

Examples

```
data <- c(rnorm(100), rnorm(100, mean = 2))
result <- detect_regimes(data)
plot(result)
plot(result, type = "segments")
```

plot_compare *Plot Multiple Results for Comparison*

Description

Creates a faceted plot comparing results from different methods.

Usage

```
plot_compare(..., names = NULL)
```

Arguments

...	Multiple regime_result objects
names	Names for each result

Value

A ggplot object

plot_interactive *Create Interactive Plot*

Description

Creates an interactive version of the changepoint plot using plotly.

Usage

```
plot_interactive(result, ...)
```

Arguments

result	A regime_result object
...	Additional arguments

Value

A plotly object

plot_summary *Create Multi-Panel Summary Plot*

Description

Creates a comprehensive multi-panel visualization of detection results.

Usage

```
plot_summary(result, ...)
```

Arguments

result	A regime_result object
...	Additional arguments

Value

A combined ggplot object

poisson_gamma *Gamma-Poisson Prior for Count Data*

Description

Creates a Gamma prior specification for Poisson-distributed count data.

Usage

```
poisson_gamma(alpha0 = 1, beta0 = 1)
```

Arguments

alpha0	Shape parameter for the rate
beta0	Rate parameter

Value

An object of class "regime_prior"

Examples

```
prior <- poisson_gamma(alpha0 = 1, beta0 = 1)
```

precision_score *Precision Score with Tolerance*

Description

Proportion of detected changepoints that are within tolerance of a true one.

Usage

```
precision_score(detected, true_cp, tolerance = 5)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
tolerance	Maximum distance for a match

Value

Precision (0 to 1)

rand_index *Rand Index for Segmentation*

Description

Measures agreement between detected and true segmentations. Based on pairwise comparisons of whether points are in the same segment.

Usage

```
rand_index(detected, true_cp, n)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
n	Total number of observations

Value

Rand Index (0 to 1)

recall_score *Recall Score with Tolerance*

Description

Proportion of true changepoints that are matched by a detection.

Usage

```
recall_score(detected, true_cp, tolerance = 5)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations
tolerance	Maximum distance for a match

Value

Recall (0 to 1)

regime_detector *Create Online Regime Detector*

Description

Creates a detector object for online (sequential) changepoint detection. The detector maintains state and can be updated incrementally as new observations arrive.

Usage

```
regime_detector(
  method = c("bocpd", "cusum", "shiryaev"),
  prior = NULL,
  hazard = NULL,
  threshold = NULL,
  ...
)
```

Arguments

method	Detection method: "bocpd", "cusum", or "shiryaev"
prior	Prior specification for Bayesian methods
hazard	Hazard prior for changepoint occurrence
threshold	Detection threshold (probability or statistic value)
...	Additional method-specific parameters

Value

An object of class "regime_detector"

Examples

```
detector <- regime_detector(method = "bocpd",
                            prior = normal_gamma(),
                            threshold = 0.5)


for (x in rnorm(100)) {
  detector <- update(detector, x)
  if (detector$last_result$alarm) {
    message("Change detected at observation ", detector$last_result$t)
    detector <- reset(detector)
  }
}
```

---

reset *Reset a detector to its initial state*

Description

Reset a detector to its initial state

Usage

```
reset(object, ...)

## S3 method for class 'regime_detector'
reset(object, ...)
```

Arguments

object	A detector object (e.g., from BOCPD or Shiryaev-Roberts)
...	Additional arguments

Value

The reset detector object

Methods (by class)

- reset(regime_detector): Reset method for regime_detector

rmse_changepoints *RMSE for Changepoints*

Description

RMSE for Changepoints

Usage

```
rmse_changepoints(detected, true_cp)
```

Arguments

detected	Vector of detected changepoint locations
true_cp	Vector of true changepoint locations

Value

Root mean squared error

shiryaev_roberts *Shiryaev-Roberts Changepoint Detection*

Description

Implements the Shiryaev-Roberts procedure, which is asymptotically optimal for detecting changes with minimal detection delay.

Usage

```
shiryaev_roberts(
  data,
  type = "mean",
  prior = NULL,
  hazard = NULL,
  threshold = 100,
  mu0 = NULL,
  mu1 = NULL,
  sigma = NULL,
  ...
)
```

Arguments

data	Numeric vector
type	Type of change to detect
prior	Prior specification
hazard	Hazard prior
threshold	Detection threshold
mu0	Pre-change mean (if known)
mu1	Post-change mean (if known)
sigma	Known standard deviation (if applicable)
...	Additional arguments

Value

List with changepoints and statistics

References

Shiryaev, A. N. (1963). On Optimum Methods in Quickest Detection Problems. Theory of Probability and Its Applications.

Examples

```
data <- c(rnorm(100), rnorm(100, mean = 1))
result <- shiryaev_roberts(data)
```

---

simulated_changepoints

Simulated Changepoints Benchmark Dataset

Description

A collection of simulated datasets with known changepoints for benchmarking changepoint detection methods.

Usage

```
simulated_changepoints
```

Format

A list with multiple scenarios:

single_mean Single mean change from 0 to 3

single_variance Single variance change from 1 to 9

multiple_mean Three mean changes

gradual Gradual trend change (challenging)

small_change Small mean change of 0.5 SD (challenging)

close_changepoints Three closely spaced changepoints (challenging)

heavy_tailed Mean change with t-distributed noise

multivariate Bivariate mean and covariance change

Each scenario contains:

- `data`: The time series data
- `true_changepoints`: Vector of true changepoint locations
- `type`: Type of change (mean, variance, etc.)
- `description`: Description of the scenario

Source

Simulated data for package benchmarking

Examples

```
data(simulated_changepoints)

# Run benchmark on single mean scenario
scenario <- simulated_changepoints$single_mean
result <- detect_regimes(scenario$data, method = "pelt")
evaluate(result, true_changepoints = scenario$true_changepoints)

# Compare multiple methods
comparison <- compare_methods(
  data = scenario$data,
  methods = c("pelt", "bocpd", "binseg"),
  true_changepoints = scenario$true_changepoints
)
print(comparison)
```

sparse_projection_cpd *Sparse Projection Changepoint Detection*

Description

Detects changepoints in high-dimensional data using random sparse projections to reduce dimensionality while preserving changepoint structure.

Usage

```
sparse_projection_cpd(
  data,
  n_projections = 10,
  penalty = "bic",
  min_segment = 5,
  sparsity = 0.3
)
```

Arguments

data	Matrix with rows = observations, columns = dimensions
n_projections	Number of random projections (default: 10)
penalty	Penalty for changepoints
min_segment	Minimum segment length
sparsity	Sparsity level for projections (fraction of non-zeros)

Value

List with changepoints and projection information

References

Wang, D. and Bhattacharjee, M. (2021). High-dimensional changepoint estimation via sparse projection. arXiv preprint.

tcn_detect *TCN-based Changepoint Detection*

Description

Uses Temporal Convolutional Networks with dilated causal convolutions for sequence-to-sequence changepoint prediction.

Usage

```
tcn_detect(
  data,
  true_changepoints = NULL,
  window_size = 64,
  n_filters = 64,
  kernel_size = 3,
  dilations = c(1, 2, 4, 8, 16),
  dropout = 0.2,
  epochs = 50,
  threshold = 0.5,
  verbose = FALSE
)
```

Arguments

data	Numeric vector of time series data
true_changepoints	Optional vector of known changepoints for supervised training. If NULL, uses unsupervised approach.
window_size	Size of input window (default: 64)
n_filters	Number of convolutional filters (default: 64)
kernel_size	Kernel size for convolutions (default: 3)
dilations	Dilation rates (default: c(1, 2, 4, 8, 16))
dropout	Dropout rate (default: 0.2)
epochs	Training epochs (default: 50)
threshold	Detection threshold (default: 0.5)
verbose	Show training progress (default: FALSE)

Value

List with changepoints, probabilities, and model

References

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

`transformer_detect` *Transformer-based Changepoint Detection*

Description

Implements a transformer architecture inspired by TCDformer for time series changepoint detection using self-attention mechanisms.

Usage

```
transformer_detect(
  data,
  true_changepoints = NULL,
  window_size = 128,
  d_model = 64,
  n_heads = 4,
  n_layers = 2,
  d_ff = 256,
  dropout = 0.1,
  epochs = 50,
  threshold = 0.5,
  verbose = FALSE
)
```

Arguments

`data`	Numeric vector of time series data
`true_changepoints`	Optional vector of known changepoints
`window_size`	Input window size (default: 128)
`d_model`	Model dimension (default: 64)
`n_heads`	Number of attention heads (default: 4)
`n_layers`	Number of transformer layers (default: 2)
`d_ff`	Feed-forward dimension (default: 256)
`dropout`	Dropout rate (default: 0.1)
`epochs`	Training epochs (default: 50)
`threshold`	Detection threshold (default: 0.5)
`verbose`	Show progress (default: FALSE)

Value

List with changepoints, attention weights, and model

References

Wu, H., et al. (2023). TimesNet: Temporal 2D-Variation Modeling

Zhou, H., et al. (2021). Informer: Efficient Transformer for Long Sequence Time-Series Forecasting

---

`update.regime_detector` *Update Online Detector with New Observation*

Description

Update Online Detector with New Observation

Usage

```
## S3 method for class 'regime_detector'
update(object, x, ...)
```

Arguments

`object`	A regime_detector object
`x`	New observation (scalar or vector for multivariate)
`...`	Additional arguments

Value

The updated regime_detector object with results in `$last_result`

`well_log` *Well Log Dataset*

Description

A simulated well-log porosity dataset with abrupt lithology changes typical of geological formations.

Usage

```
well_log
```

Format

A numeric vector of length 1000 with attributes:

true_changepoints Vector of true changepoint locations: c(200, 350, 500, 700, 850)

description Description of the dataset

lithologies Names of lithology units

Details

The data simulates porosity measurements from a well log with six distinct lithological units:

- Unit 1 (1-200): Sandstone - porosity ~15%
- Unit 2 (201-350): Shale - porosity ~8%
- Unit 3 (351-500): Limestone - porosity ~20%
- Unit 4 (501-700): Sandstone - porosity ~12%
- Unit 5 (701-850): Dense Shale - porosity ~5%
- Unit 6 (851-1000): Sandstone - porosity ~18%

Source

Simulated data based on typical well-log characteristics

Examples

```
data(well_log)
result <- detect_regimes(well_log, method = "pelt", min_segment = 50)
plot(result, type = "segments")

# Compare with true lithology boundaries
true_cps <- attr(well_log, "true_changepoints")
evaluate(result, true_changepoints = true_cps)
```

`wild_binary_segmentation` *Wild Binary Segmentation*

Description

Detects multiple changepoints using the Wild Binary Segmentation algorithm. Uses random intervals to improve detection in long time series.

Usage

```
wild_binary_segmentation(
  data,
  type = "both",
  penalty = "BIC",
  min_segment = 2,
  n_changepoints = "multiple",
  M = 5000,
  threshold = NULL,
  ...
)
```

Arguments

data	Numeric vector or matrix of time series data
type	Type of change to detect: "mean", "variance", or "both"
penalty	Penalty for model complexity: "BIC", "AIC", "MBIC", or numeric
min_segment	Minimum segment length
n_changepoints	Expected number of changepoints: "single", "multiple", or integer
M	Number of random intervals to draw
threshold	Detection threshold for CUSUM statistic. If NULL, automatically determined
...	Additional arguments

Value

A list with:

changepoints	Vector of detected changepoint locations
n_changepoints	Number of changepoints detected
information_criterion	BIC value for the segmentation

References

Fryzlewicz, P. (2014). Wild Binary Segmentation for multiple change-point detection. Annals of Statistics, 42(6), 2243-2281.

Examples

```
data <- c(rnorm(100), rnorm(100, mean = 2), rnorm(100))
result <- wild_binary_segmentation(data)
```

Index