Package ‘focus’

June 9, 2026

Type Package

Title Online Changepoint Detection in Univariate and Multivariate Data Streams

Version 0.1.6

Date 2026-06-08

Maintainer Gaetano Romano <g.romano@lancaster.ac.uk>

Description
Provides high-performance online changepoint detection in univariate and multivariate data streams. Implements efficient 'C++' backends for the 'focus', 'md-focus' and 'np-focus' algorithms, with an 'R' interface for real-time monitoring and offline analysis.
The package bundles code from 'Qhull' <http://www.qhull.org/>, by C. B. Barber and The Geometry Center. See 'inst/COPYRIGHTS' for details.

License GPL (>= 3)

Imports Rcpp (>= 1.1.0)

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Author Gaetano Romano [aut, cre, trl],
Kes Ward [aut],
Yuntang Fan [aut],
Guillem Rigaill [aut],
Vincent Runge [aut],
Paul Fearnhead [aut],
Idris A. Eckley [aut],
C. B. Barber [ctb, cph] (Author and copyright holder of bundled 'Qhull' library),
The Geometry Center [cph] (Copyright holder of bundled 'Qhull' library)

Repository CRAN

Date/Publication 2026-06-09 10:10:02 UTC

Contents

`focus-package` *Online Changepoint Detection in Univariate and Multivariate Data Streams*

Description

The **focus** package provides high-performance implementations of online and offline changepoint detection algorithms for univariate and multivariate data streams. The package exposes optimized C++ backends (FOCuS and md-FOCuS style algorithms) with R interfaces for both real-time monitoring (sequential updates) and batch/offline analysis.

Details

focus implements efficient changepoint detection for a range of statistical models and use-cases. The package supports multiple distributional families (Gaussian, Poisson, Bernoulli, Gamma) and a non-parametric NPFOCuS variant. It provides two primary modes:

Offline mode (`focus_offline`) Processes all observations in C++ for maximum efficiency. Useful for benchmarking, for computing full statistic trajectories, or for batch processing. By default the offline call stops when a threshold is exceeded; use `threshold = Inf` to compute statistics for all observations.

Online / sequential mode (`detector_create`, `detector_update`, `get_statistics`) Create an online detector object and update it one observation at a time from R. This provides a flexible streaming API and access to candidate segments, at the cost of more R/C calls per observation.

Main features:

- Multiple distributions: Gaussian, Poisson, Bernoulli, Gamma, and non-parametric NPFOCuS.
- Univariate and multivariate observations.
- Known or unknown pre-change parameters (generalised likelihood-ratio and Page–CUSUM style tests).
- One-sided and two-sided detection.

- Flexible interface: statistical cost functions are independent from the detector candidate-management strategy.
- High-performance C++ backend designed for integration with R and direct use from C++ projects.

Important functions:

- `focus_offline(Y, threshold, type, family, ...)`: run complete offline detection; returns full statistic trajectories, detected changepoints, candidate segments, thresholds and metadata.
- `detector_create(type, ...)`: construct an online detector object.
- `detector_update(det, y)`: update the detector with a new observation.
- `get_statistics(det, family, theta0 = NULL, shape = NULL)`: compute statistics for the current detector state.
- Inspection utilities: `detector_info_n()`, `detector_info_sn()`, `detector_pieces_len()`, `detector_candidates()`.

Author(s)

Gaetano Romano [aut, cre], <g.romano@lancaster.ac.uk>\ Kes Ward [aut], <k.ward4@lancaster.ac.uk>\ Yuntang Fan [aut], <y.yuntang@lancaster.ac.uk>\ Guillem Rigaill [aut], <guillem.rigaill@inrae.fr>\ Vincent Runge [aut], <vincent.runge@univ-evry.fr>\ Paul Fearnhead [aut], <p.fearnhead@lancaster.ac.uk>\ Idris A. Eckley [aut], <i.eckley@lancaster.ac.uk>\ Maintainer: Gaetano Romano <g.romano@lancaster.ac.uk>

References

Pishchagina, L., G. Romano, P. Fearnhead, V. Runge, and G. Rigaill (2025). Online Multivariate Changepoint Detection: Leveraging Links with Computational Geometry. *JRSS B*. doi:10.1093/jrsssb/qkaf046.

Romano, G., I. A. Eckley, and P. Fearnhead (2024). A Log-Linear Nonparametric Online Changepoint Detection Algorithm Based on Functional Pruning. *IEEE Transactions on Signal Processing*, 72.

Romano, G., I. A. Eckley, P. Fearnhead, and G. Rigaill (2023). Fast Online Changepoint Detection via Functional Pruning CUSUM Statistics. *Journal of Machine Learning Research*, 24(81):1–36.

Ward, K., G. Romano, I. Eckley, and P. Fearnhead (2024). A Constant-Per-Iteration Likelihood Ratio Test for Online Changepoint Detection for Exponential Family Models. *Statistics and Computing*, 34(3).

Note: The package bundles and links to code from Qhull (C. B. Barber et al., The Geometry Center). Qhull functions are used for convex-hull computations in multivariate pruning. See '`inst/COPYRIGHTS/`' for the Qhull license and attribution details.

See Also

`focus_offline`, `detector_create`, `detector_update`, `get_statistics`, `detector_candidates`, `generate_projection_indexes`

Examples

```
## Offline (batch) example: univariate Gaussian change-in-mean
set.seed(123)
Y <- c(rnorm(500, mean = 0), rnorm(500, mean = 2))

# Run offline detection (C++ loop). Use threshold=Inf to compute full trajectory.
res <- focus_offline(Y, threshold = 20, type = "univariate", family = "gaussian")
if (!is.null(res$detection_time)) {
  cat("Detection at time:", res$detection_time, "\n")
}

## Online (sequential) example
det <- detector_create(type = "univariate")
stat_trace <- numeric(length(Y))
threshold <- 20
for (i in seq_along(Y)) {
  detector_update(det, Y[i])
  r <- get_statistics(det, family = "gaussian")
  stat_trace[i] <- r$stat
  if (!is.null(r$stat) && r$stat > threshold) {
    cat("Online detection at", i, "estimate tau =", r$changepoint, "\n")
    break
  }
}

## Multivariate offline example (p = 3)
set.seed(42)
p <- 3
Y_multi <- rbind(
  matrix(rnorm(1000 * p, mean = 0), ncol = p),
  matrix(rnorm(500 * p, mean = 1.2), ncol = p)
)
res_multi <- focus_offline(Y_multi, threshold = 30, type = "multivariate", family = "gaussian")
cat("Multivariate detection time:", res_multi$detection_time, "\n")
```

`detector_candidates` *Get candidate segments*

Description

Returns detailed information about all candidate changepoint segments currently tracked by the detector.

Usage

```
detector_candidates(det_ptr)
```

Arguments

det_ptr	External pointer to detector created by detector_create().

Details

Each row represents a candidate segment from time tau to the current time. The sufficient statistics in st are used to efficiently compute test statistics without reprocessing the data.

Value

A data frame (tibble) with columns:

tau	Integer vector. Candidate changepoint locations (0-based indices).
st	List of numeric vectors. Sufficient statistics for each candidate segment (e.g., cumulative sums of the data).
side	Character vector. Side indicator for each candidate (relevant for one-sided detectors).

---

detector_create *Create a FOCuS changepoint detector*

Description

Creates an online (sequential) changepoint detector object that provides a step-by-step interface to the FOCuS algorithm. Each call to detector_update() adds new data, and get_statistics() computes the current test statistic and detection result.

Usage

```
detector_create(
  type,
  dim_indexes = NULL,
  quantiles = NULL,
  pruning_mult = 2L,
  pruning_offset = 1L,
  side = "right",
  anomaly_intensity = NULL,
  rho = NULL,
  mu0_arp = NULL
)
```

Arguments

type	Character string specifying detector type. One of: • "univariate": Two-sided univariate detection • "univariate_one_sided": One-sided univariate detection • "multivariate": Multivariate detection with projections • "npfocus": Nonparametric detection (NP-FOCuS). See details. • "arp": AutoRegressive Process detection. Requires rho parameter.
dim_indexes	List of integer vectors specifying projection index sets for high-dimensional multivariate detectors. Not required for sequences of dimentions less than 5. Each element is a vector of 0-based column indices. Default is NULL.
quantiles	Numeric vector of quantiles for nonparametric ("npfocus") detectors. Required when type = "npfocus". Default is NULL.
pruning_mult	Integer. Candidate pruning multiplier parameter. Default is 2.
pruning_offset	Integer. Candidate pruning offset parameter. Default is 1.
side	Character string. For one-sided detectors, either "right" (detects increases) or "left" (detects decreases). Default is "right".
anomaly_intensity	Numeric scalar. Anomaly intensity threshold for pruning candidates. Only candidates with sufficient signal magnitude are retained. Default is NULL (disabled).
rho	Numeric vector. AR coefficients for AutoRegressive Process (ARP) detectors. Required when type = "arp". Default is NULL.
mu0_arp	Numeric scalar. Pre-change mean for ARP detectors (optional). When provided, enables more efficient pruning by filtering candidates based on the known pre-change parameter. Only used when type = "arp". Default is NULL.

Details

The detector maintains sufficient statistics internally and uses pruning to efficiently track candidate changepoints. The pruning_mult and pruning_offset parameters control the pruning strategy.

AutoRegressive Process (ARP): When type = "arp", the rho parameter must be provided as a numeric vector of AR coefficients (lag-1, lag-2, ..., lag-p). The detector then computes statistics optimal for detecting changepoints in AR(p) processes. Use get_statistics(family = "arp") to retrieve the test statistics. The optional theta0 parameter specifies the pre-change mean and is tied to the pruning logic: if provided, it enables more efficient pruning by allowing the algorithm to filter candidates based on the known pre-change parameter.

High-dimentional multivariate detectors: For high-dimensional multivariate detection, computing the full hull would be too prohibitive. Additionally, the complexity is expected to be log(n)^p, where n is the number of iterations and p is the dimensions. So for short, high-dimentional sequences, it is possible to reconstruct the set of changepoint locations by approximating the hull on projections in smaller dimentions. dim_indexes specifies which dimensions to use for each projection of the convex hull for pruning. Use generate_projection_indexes() to generate systematic projection sets.

NPFOCuS: For non-parametric detection one needs to set detector(type = "npfocus") and the cost can be computed as get_statistics(family = "npfocus"). Any other cost will not work with this detector type. The quantiles vector argument is required, see get_statistics() for details.

Value

An external pointer (SEXP) to the detector object. This should be passed to other detector functions like `detector_update()` and `get_statistics()`.

Examples

```
# Univariate detector
det <- detector_create(type = "univariate")
detector_update(det, 0.5)
detector_update(det, 1.2)
r <- get_statistics(det, family = "gaussian")
print(r)

## Online (sequential) example
# Generate data with a changepoint
set.seed(123)
Y <- c(rnorm(500, mean = 0), rnorm(500, mean = 1))
det <- detector_create(type = "univariate")
stat_trace <- numeric(length(Y))
threshold <- 20
for (i in seq_along(Y)) {
  detector_update(det, Y[i])
  r <- get_statistics(det, family = "gaussian")
  stat_trace[i] <- r$stat
  if (!is.null(r$stat) && r$stat > threshold) {
    cat("Online detection at", i, "estimate tau =", r$changepoint, "\n")
    plot(stat_trace[1:i], type = "l", ylab = "Test Statistic", xlab = "Time")
    break
  }
}

# Multivariate detector with projections
dim_indexes <- list(c(0,1), c(1,2), c(0,2))  # 0-based indices
det_mv <- detector_create(type = "multivariate", dim_indexes = dim_indexes)
detector_update(det_mv, c(0.5, 1.2, -0.3))

# Nonparametric detector
quants <- qnorm(c(0.25, 0.5, 0.75))
det_np <- detector_create(type = "npfocus", quantiles = quants)

# One-sided univariate detector
det_one_sided <- detector_create(type = "univariate_one_sided", side = "left")
```

`detector_info_n` *Get number of observations processed*

Description

Returns the total number of observations processed by the detector.

Usage

```
detector_info_n(det_ptr)
```

Arguments

det_ptr	External pointer to detector created by detector_create().

Value

Integer. Number of observations processed (current time index).

---

detector_info_sn *Get cumulative sum statistic*

Description

Returns the current cumulative sum statistic maintained by the detector.

Usage

```
detector_info_sn(det_ptr)
```

Arguments

det_ptr	External pointer to detector created by detector_create().

Value

Numeric vector. Cumulative sum statistic. For univariate detectors, a scalar (length-1 vector). For multivariate detectors, a vector of length equal to the number of dimensions.

---

detector_pieces_len *Get number of candidate segments*

Description

Returns the number of candidate changepoint segments currently tracked by the detector.

Usage

```
detector_pieces_len(det_ptr)
```

Arguments

det_ptr	External pointer to detector created by detector_create().

Details

The FOCuS algorithm maintains a set of candidate segments that could potentially contain change-points. This number grows with time but is controlled by the pruning parameters.

Value

Integer. Number of candidate segments.

`detector_update` *Update detector with new observation(s)*

Description

Adds new observation(s) to the detector's internal state and updates sufficient statistics.

Usage

```
detector_update(det_ptr, y, lambda = 1)
```

Arguments

`det_ptr`	External pointer to detector created by `detector_create()`.
`y`	Numeric vector of new observation(s). For univariate detectors, this should be a scalar (length-1 vector). For multivariate detectors, this should be a vector matching the number of dimensions.
`lambda`	Numeric scalar. Rate parameter for background process (default: 1.0). Allows for non-fixed background rate. For example, use `lambda_i` for observation-specific rates. Default is `1.0` (standard CUSUM, one observation per update).

Value

An external pointer to the detector (the same object that was passed in). The detector is updated *in place* — no copy is made — so this return value is provided only for convenience, for example when using the native R pipe operator ('|>') e.g., `det |> detector_update(y) |> get_statistics()`.

Examples

```
# Univariate example
det <- detector_create(type = "univariate")
detector_update(det, 0.5)
detector_update(det, 1.2)

# Multivariate example
det_mv <- detector_create(type = "multivariate")
detector_update(det_mv, c(0.5, 1.2, -0.3))
```

```
## Online (sequential) example
# Generate data with a changepoint
set.seed(123)
Y <- c(rnorm(500, mean = 0), rnorm(500, mean = 1))
det <- detector_create(type = "univariate")
stat_trace <- numeric(length(Y))
threshold <- 20
for (i in seq_along(Y)) {
  detector_update(det, Y[i])
  r <- get_statistics(det, family = "gaussian")
  stat_trace[i] <- r$stat
  if (!is.null(r$stat) && r$stat > threshold) {
    cat("Online detection at", i, "estimate tau =", r$changepoint, "\n")
    plot(stat_trace[1:i], type = "l", ylab = "Test Statistic", xlab = "Time")
    break
  }
}
```

`focus_offline` *Run FOCuS detector in offline batch mode*

Description

Processes all data at once and returns detection results and trajectories. This is the most efficient way to run changepoint detection when all data is available upfront.

Usage

```
focus_offline(
  Y,
  threshold,
  type = "univariate",
  family = "gaussian",
  theta0 = NULL,
  dim_indexes = NULL,
  quantiles = NULL,
  pruning_mult = 2L,
  pruning_offset = 1L,
  side = "right",
  shape = NULL,
  anomaly_intensity = NULL,
  rho = NULL,
  mu0_arp = NULL
)
```

Arguments

Y	Numeric vector or matrix. Data array. For univariate detection, a numeric vector. For multivariate detection, a matrix with observations in rows and dimensions in columns.
threshold	Numeric scalar or vector. Detection threshold(s). Can be: • A scalar: applied to all test statistics (default behavior for most cost functions) • A vector: length must match the number of test statistics (for example, npfocus returns both the sum and max statistics, so threshold should be length 2). • Inf: no thresholding (returns all statistics)
type	Character string specifying detector type. See detector_create() for options. Defaults to "univariate".
family	Character string specifying distribution family. See get_statistics() for options. Defaults to "gaussian".
theta0	Numeric vector. Null hypothesis parameter. Default is NULL.
dim_indexes	List of integer vectors. Projection index sets for multivariate detectors. Default is NULL.
quantiles	Numeric vector. Quantiles for nonparametric detectors. Default is NULL.
pruning_mult	Integer. Pruning multiplier parameter. Default is 2.
pruning_offset	Integer. Pruning offset parameter. Default is 1.
side	Character string. For one-sided detectors: "right" or "left". Default is "right".
shape	Numeric scalar. Shape parameter for gamma distribution. Default is NULL.
anomaly_intensity	Numeric scalar. Anomaly intensity threshold for pruning candidates. Only candidates with sufficient signal magnitude are retained. Default is NULL (disabled).
rho	Numeric vector. AR coefficients for AutoRegressive Process (ARP) detectors. Required when type = "arp". Default is NULL.
mu0_arp	Numeric scalar. Pre-change mean for ARP detectors (optional). Only used when type = "arp". Default is NULL.

Details

This function runs the complete detection algorithm in C++ for maximum efficiency. It processes observations sequentially and stops at the first detection (when any statistic exceeds its threshold).

For multivariate data, the algorithm computes multiple statistics (one per projection). Detection occurs when ANY statistic exceeds its threshold.

Value

A list with components:

stat	Numeric matrix. Test statistics over time (n_obs × n_stats). Each row corresponds to one time point, each column to one statistic.
changepoint	Integer vector. Detected changepoints at each time point (1-based indices), or NA if no changepoint detected at that time.
detection_time	Integer or NULL. Time of first detection (1-based), or NULL if no detection occurred.
detected_changepoint	Integer or NULL. Changepoint location at detection time (1-based), or NULL if no detection occurred.
candidates	Data frame. Final candidate segments (see detector_candidates()).
threshold	Numeric vector. Threshold(s) used for detection.
n	Integer. Number of observations processed.
type	Character. Detector type used.
family	Character. Distribution family used.
shape	Numeric or NULL. Shape parameter (for gamma family).

Examples

```
# Univariate Gaussian detection
set.seed(123)
Y <- c(rnorm(100, mean = 0), rnorm(100, mean = 2))
result <- focus_offline(Y, threshold = 10, type = "univariate",
                        family = "gaussian")
cat("Detection at time:", result$detection_time, "\n")
cat("Changepoint at:", result$detected_changepoint, "\n")

# Plot statistics
plot(result$stat, type = "l", ylab = "Test Statistic", xlab = "Time")
abline(h = result$threshold, col = "red", lty = 2)
if (!is.null(result$detection_time)) {
  abline(v = result$detection_time, col = "blue", lty = 2)
}

# Poisson detection
Y_poisson <- c(rpois(100, lambda = 2), rpois(100, lambda = 5))
result_poisson <- focus_offline(Y_poisson, threshold = 10,
                                type = "univariate",
                                family = "poisson")
```

---

generate_projection_indexes

Generate projection index sets

Description

Generates projection index sets for high-dimensional multivariate detectors using circular combinations.

Usage

```
generate_projection_indexes(d, p)
```

Arguments

d	Integer. Total number of dimensions.
p	Integer. Projection subset size (number of dimensions per projection).

Details

This function generates systematic projection sets for use with multivariate detectors. The circular combination approach ensures good coverage of the dimensional space while keeping the number of projections manageable.

Value

A list of integer vectors. Each element is a vector of 0-based column indices representing one projection.

Examples

```
# Generate 2-dimensional projections from 5 dimensions
proj <- generate_projection_indexes(d = 5, p = 2)
print(proj)

# Use with multivariate detector
det <- detector_create(type = "multivariate", dim_indexes = proj)
set.seed(42)
d <- 5

# Create data: changepoint at t=1000
Y_multi <- rbind(
    matrix(rnorm(1000 * d, mean = -1, 1), ncol = d),
    matrix(rnorm(500 * d, mean = 1.2), ncol = d)
)

# Full multivariate detection
system.time(
res_multi <- focus_offline(Y_multi, threshold = Inf,
                           type = "multivariate", family = "gaussian")
)

# Low-dimensional projection approximation
dim_indexes <- generate_projection_indexes(5, 2)
system.time(
res_multi_approx <- focus_offline(Y_multi, threshold = Inf,
                                  type = "multivariate", family = "gaussian",
                                  dim_indexes = dim_indexes)
)

# Verify similarity
```

```
all.equal(res_multi$stat, res_multi_approx$stat)
```

`get_statistics` *Compute current changepoint statistics*

Description

Computes the current changepoint test statistic and detection result based on all observations processed so far.

Usage

```
get_statistics(det_ptr, family, theta0 = NULL, shape = NULL)
```

Arguments

`det_ptr`	External pointer to detector created by `detector_create()`.
`family`	Character string specifying the distribution family: • `"gaussian"`: Gaussian (normal) distribution • `"poisson"`: Poisson distribution • `"bernoulli"`: Bernoulli (binary) distribution • `"gamma"`: Gamma distribution (requires `shape` parameter) • `"npfocus"`: Nonparametric detection (see details) • `"arp"`: AutoRegressive Process detection (requires detector created with `type = "arp"`)
`theta0`	Numeric vector specifying the null hypothesis parameter. For univariate detectors: scalar (length-1 vector). For multivariate detectors: vector matching the number of dimensions. Default is `NULL`.
`shape`	Numeric scalar. Shape parameter for gamma distribution. Required and must be positive when `family = "gamma"`. Default is `NULL`.

Details

The function computes a log-likelihood ratio test statistic comparing the null hypothesis (no change) against the alternative (a change at the optimal location). The statistic is typically compared against a threshold to determine if a changepoint should be declared.

Gamma family: When `family = "gamma"` a positive `shape` parameter must be provided; otherwise an error is raised. Passing `shape` for a non-gamma family will be ignored (and in some interfaces will trigger a warning).

NPFOCuS: For non-parametric detection one needs to set `detector(type = "npfocus")` and the cost can be computed as `get_statistics(family = "npfocus")`. The `quantiles` vector argument is required. NPFOCuS returns two statistics (sum and max over quantiles) as a vector; in the offline interface `stat` will be a matrix with two columns.

AutoRegressive Process (ARP): For ARP detection, use `family = "arp"` with a detector created via `detector_create(type = "arp", rho = ...)`. The AR coefficients (rho) are already built into the detector at creation time. The optional `theta0` parameter specifies the pre-change mean (if known) and is used for pruning logic; if not provided (`NULL`), pruning operates without this information. Returns a scalar test statistic optimized for detecting changepoints in AR processes.

Value

A list with components:

`stopping_time`	Integer. Current time index (number of observations processed).
`changepoint`	Integer or NULL. Detected changepoint location (1-based index), or NULL if no changepoint detected.
`stat`	Numeric scalar or vector. Test statistic(s). For univariate detectors, a scalar. For multivariate detectors, a vector of statistics for each projection.

Examples

```
## Online (sequential) example
# Generate data with a changepoint
set.seed(123)
Y <- c(rnorm(500, mean = 0), rnorm(500, mean = 1))
det <- detector_create(type = "univariate")
stat_trace <- numeric(length(Y))
threshold <- 20
for (i in seq_along(Y)) {
  detector_update(det, Y[i])
  r <- get_statistics(det, family = "gaussian")
  stat_trace[i] <- r$stat
  if (!is.null(r$stat) && r$stat > threshold) {
    cat("Online detection at", i, "estimate tau =", r$changepoint, "\n")
    plot(stat_trace[1:i], type = "l", ylab = "Test Statistic", xlab = "Time")
    break
  }
}

## Note that multiple models can be tested simultaneously on the same detector
# as the statistics is independent of the detector state.
# For example, testing both Gaussian and Poisson costs
set.seed(2024)
# Generate Poisson count data with a rate change
Y_counts <- c(rpois(500, lambda = 10), rpois(500, lambda = 15))
# Compute full trajectories for comparison
det2 <- detector_create(type = "univariate")
stat_gaussian <- numeric(length(Y_counts))
stat_poisson <- numeric(length(Y_counts))

for (i in seq_along(Y_counts)) {
  detector_update(det2, Y_counts[i])
  stat_gaussian[i] <- get_statistics(det2, family = "gaussian")$stat
  stat_poisson[i] <- get_statistics(det2, family = "poisson", theta0 = 10)$stat
}
```

```
# Plot comparison
oldpar <- par(mfrow = c(1, 2))
plot(stat_gaussian, type = "l", main = "Gaussian Statistic on Poisson Data",
     xlab = "Time", ylab = "Statistic", lwd = 2, col = "blue")
abline(v = 500, col = "green", lty = 3, lwd = 2)

plot(stat_poisson, type = "l", main = "Poisson Statistic on Poisson Data",
     xlab = "Time", ylab = "Statistic", lwd = 2, col = "red")
abline(v = 500, col = "green", lty = 3, lwd = 2)
par(oldpar)
```

Index