Package: focus (via r-universe)

November 13, 2026

Type Package

Title Online Changepoint Detection in Univariate and Multivariate Data Streams

Version 0.1.11

Date 2026-10-14

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)

Depends R (>= 3.6.0)

Imports graphics, Rcpp (>= 1.1.0)

LinkingTo Rcpp

Encoding UTF-8

Suggests ggplot2, testthat (>= 3.0.0)

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Gaetano Romano [aut, cre, trl], Kes Ward [aut], Yuntang Fan [aut], Guillem Rigaill [aut], Vincent Runge [aut], Idris A. Eckley [aut], Paul Fearnhead [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 https://gtromano.r-universe.dev

Date/Publication 2026-10-14 12:07:04 UTC

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

RemoteRef HEAD

RemoteSha 4628fc0e6c859873699c2f746583a6b23b5e33c7

Contents

focus-package . . . 2
detector_candidates . . . 5
detector_cands_len . . . 6
detector_create . . . 7
detector_info_n . . . 9
detector_info_sn . . . 10
detector_update . . . 11
focus-methods . . . 12
focus_offline . . . 15
generate_projection_indexes . . . 17
get_statistics . . . 19

Index **22**

`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_cands_len()`, `detector_candidates()`.

The detector, the statistics, the offline results and the projection index sets are S3 objects of class `"focus_detector"`, `"focus_statistics"`, `"focus_offline"` and `"focus_projections"`, respectively, with `print` methods and, for detectors and offline results, `summary` methods; offline results also have a `plot` method. See `focus-methods`.

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

```
## Univariate Gaussian change in mean
set.seed(123)
Y <- c(rnorm(500, mean = 0), rnorm(500, mean = 2))

## 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
  }
}
summary(det, family = "gaussian")

## Offline (batch) example on the same data
# 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")
}
summary(res)
plot(res, data = Y)

## Multivariate detection
set.seed(123)
d <- 3
Y_multi <- rbind(
  matrix(rnorm(300 * d, mean = 0), ncol = d),
  matrix(rnorm(100 * d, mean = 1), ncol = d)
)
det_mv <- detector_create(type = "multivariate")
for (i in seq_len(nrow(Y_multi))) {
  detector_update(det_mv, Y_multi[i, ])
  r <- get_statistics(det_mv, family = "gaussian")
  if (!is.null(r$stat) && r$stat > 30) {
    cat("Multivariate detection at", i, "estimate tau =", r$changepoint, "\n")
    break
  }
}
summary(det_mv)

## Non-parametric detection (NP-FOCuS)
# Heavy-tailed data with a change in location
set.seed(123)
Y_np <- c(rt(500, df = 2), rt(200, df = 2) + 1)
# Quantiles of the pre-change distribution
```

```
quants <- qt(c(0.1, 0.25, 0.5, 0.75, 0.9), df = 2)
det_np <- detector_create(type = "npfocus", quantiles = quants)
threshold_np <- c(50, 20)  # for the sum and the max statistics
for (i in seq_along(Y_np)) {
  detector_update(det_np, Y_np[i])
  r <- get_statistics(det_np, family = "npfocus")
  if (!is.null(r$stat) && any(r$stat > threshold_np)) {
    cat("Non-parametric detection at", i, "\n")
    break
  }
}
r

## Autoregressive detection: AR(3) with a change in mean
set.seed(123)
rho <- c(0.5, -0.3, 0.2)  # AR coefficients
Y_ar <- arima.sim(list(ar = rho), n = 700) + rep(c(0, 1.5), c(500, 200))
det_ar <- detector_create(type = "arp", rho = rho, mu0_arp = 0)
for (i in seq_along(Y_ar)) {
  detector_update(det_ar, Y_ar[i])
  r <- get_statistics(det_ar, family = "arp")
  if (!is.null(r$stat) && r$stat > 30) {
    cat("AR(3) detection at", i, "estimate tau =", r$changepoint, "\n")
    break
  }
}
det_ar
```

`detector_candidates` *Get the Candidate Segments*

Description

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

Usage

```
detector_candidates(det_ptr, ...)

## S3 method for class 'focus_detector'
detector_candidates(det_ptr, ...)
```

Arguments

`det_ptr`	A `"focus_detector"` object created by `detector_create()`.
`...`	Further arguments passed to or from other methods.

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. `detector_candidates()` is an S3 generic, with a method for `"focus_detector"` objects.

Value

A data frame (tibble) with one row per candidate and columns:

`tau`	Numeric vector. Candidate changepoint locations, on the same scale as the changepoint estimate returned by `get_statistics()`.
`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).

Examples

```
set.seed(1)
det <- detector_create(type = "univariate")
for (y in rnorm(50)) detector_update(det, y)
head(detector_candidates(det))
```

`detector_cands_len` *Get the Number of Candidate Segments*

Description

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

Usage

```
detector_cands_len(det_ptr, ...)

## S3 method for class 'focus_detector'
detector_cands_len(det_ptr, ...)
```

Arguments

`det_ptr`	A `"focus_detector"` object created by `detector_create()`.
`...`	Further arguments passed to or from other methods.

Details

The FOCuS algorithm maintains a set of candidate segments that could potentially contain changepoints. This number grows with time but is controlled by the pruning parameters. `detector_cands_len()` is an S3 generic, with a method for `"focus_detector"` objects.

Value

Integer. Number of candidate segments.

Examples

```
set.seed(1)
det <- detector_create(type = "univariate")
for (y in rnorm(50)) detector_update(det, y)
detector_cands_len(det)
```

`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 data with at most 5 dimensions). 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 mu0_arp 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-dimensional multivariate detectors. For high-dimensional multivariate detection, computing the full convex hull is prohibitive, as the expected number of candidates grows as $\log(n)^p$, where n is the number of observations and p the number of dimensions. The set of candidate changepoints can then be approximated by computing the hull on lower-dimensional projections: dim_indexes specifies which dimensions to use for each projection. Use generate_projection_indexes() to generate systematic projection sets.

NPFOCuS. For non-parametric detection, create the detector with detector_create(type = "npfocus", quantiles = ...) and compute the statistics with get_statistics(family = "npfocus"). No other family can be used with this detector type.

Value

An object of class "focus_detector": an external pointer to the state of the C++ detector, which is updated in place. It should be passed to the other detector functions, such as detector_update() and get_statistics(). print and summary methods are available, see focus-methods. As external pointers, detectors cannot be saved and restored across R sessions.

Examples

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

```
det <- detector_update(det, 1.2)
det
r <- get_statistics(det, family = "gaussian")
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 the Number of Observations Processed*

Description

Returns the total number of observations processed by the detector.

Usage

```
detector_info_n(det_ptr, ...)

## S3 method for class 'focus_detector'
detector_info_n(det_ptr, ...)
```

Arguments

det_ptr	A "focus_detector" object created by detector_create().
...	Further arguments passed to or from other methods.

Details

detector_info_n() is an S3 generic, with a method for "focus_detector" objects.

Value

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

Examples

```
det <- detector_create(type = "univariate")
for (y in c(0.5, 1.2, -0.3)) detector_update(det, y)
detector_info_n(det)
```

---

detector_info_sn *Get the Cumulative Sum Statistic*

Description

Returns the current cumulative sum statistic maintained by the detector.

Usage

```
detector_info_sn(det_ptr, ...)

## S3 method for class 'focus_detector'
detector_info_sn(det_ptr, ...)
```

Arguments

det_ptr	A "focus_detector" object created by detector_create().
...	Further arguments passed to or from other methods.

Details

detector_info_sn() is an S3 generic, with a method for "focus_detector" objects.

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.

Examples

```
det <- detector_create(type = "multivariate")
detector_update(det, c(0.5, 1.2))
detector_update(det, c(-0.3, 0.4))
detector_info_sn(det)
```

`detector_update` *Update a Detector with New Observations*

Description

Adds a new observation to the detector's internal state, updates the sufficient statistics and prunes the set of candidate changepoints.

Usage

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

Arguments

`det_ptr`	A `"focus_detector"` object created by `detector_create()`.
`y`	Numeric vector with the new observation. 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. Weight of the new observation: the internal time counter is incremented by `lambda` rather than by one, which allows for a non-constant background rate (e.g., observation-specific exposures). Default is `1.0`, i.e., one observation per update.

Value

The same `"focus_detector"` object that was passed in. The detector is updated *in place* (no copy is made), so the return value is provided only for convenience, for example to chain calls with the native pipe operator, as in `det |> detector_update(y) |> get_statistics(family = "gaussian")`.

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-methods` *Methods for Detector, Statistics, Offline Result and Projection Objects*

Description

Methods for the S3 classes returned by the main functions of the package: `"focus_detector"` objects, created by `detector_create()`; `"focus_statistics"` objects, returned by `get_statistics()`; `"focus_offline"` objects, returned by `focus_offline()`; and `"focus_projections"` objects, returned by `generate_projection_indexes()`.

Usage

```
## S3 method for class 'focus_detector'
print(x, ...)

## S3 method for class 'focus_statistics'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'focus_offline'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'focus_offline'
summary(object, ...)

## S3 method for class 'summary.focus_offline'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'focus_offline'
```

```
plot(
  x,
  data = NULL,
  type = "l",
  lty = 1,
  col = NULL,
  xlab = "Time",
  ylab = "Statistic",
  main = NULL,
  ...
)

## S3 method for class 'focus_detector'
summary(object, family = NULL, theta0 = NULL, shape = NULL, ...)

## S3 method for class 'summary.focus_detector'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'focus_projections'
print(x, n = 10L, ...)

## S3 method for class 'focus_projections'
x[i]

## S3 method for class 'focus_projections'
as.matrix(x, ...)

## S3 method for class 'focus_offline'
autoplot(object, data = NULL, ...)
```

Arguments

x, object	An object of class "focus_detector", "focus_statistics", "focus_offline", "focus_projections", "summary.focus_detector" or "summary.focus_offline", as appropriate.
...	Further arguments passed to or from other methods (for plot, to matplot()).
digits	Number of significant digits used to print the statistics.
data	Optionally, the data passed to focus_offline(), drawn above the trace of the statistic(s).
type, lty, col, xlab, ylab, main	Graphical parameters passed to matplot(). By default, one solid line is drawn per test statistic.
family, theta0, shape	Optional arguments passed to get_statistics(): if family is given, the summary of a detector includes its current statistics.
n	Maximum number of projections to print.
i	Indices of the projections to keep.

Details

The `print` methods give a compact description of the object: for a detector, its type, the number of observations processed and the number of candidate changepoints currently stored; for the statistics, the current time, changepoint estimate and test statistic(s); for an offline result, the detector type and family, the threshold(s) and the detection, if any; for projection index sets, the (0-based) indices of each projection, one projection per row.

The `summary` method for `"focus_detector"` objects additionally reports the cumulative sums (see `detector_info_sn()`) and the locations of the candidate changepoints, at the time it is called, and, if a `family` is given, the current statistics.

The `summary` method for `"focus_offline"` objects additionally reports, for each test statistic, its maximum over time, the time at which the maximum is attained, the changepoint estimate at that time and the threshold.

The `plot` method for `"focus_offline"` objects draws the trace of the test statistic(s) over time, with the finite threshold(s) as dashed horizontal lines and, if a detection occurred, the estimated changepoint as a dotted vertical line. If `data` are given, they are drawn above the trace, with one panel per dimension. If **ggplot2** is installed, `autoplot()` returns the same plot as a `"ggplot"` object.

Projection index sets can be subset with `[` (e.g., with `head()`), keeping their class, and `as.matrix` returns them as a matrix with one row per projection.

Value

The `print` and `plot` methods return `x` invisibly. The `summary` method for detectors returns an object of class `"summary.focus_detector"`, a list with the reported elements. The `summary` method for offline results returns an object of class `"summary.focus_offline"`: a list with the elements `type`, `family`, `shape`, `n`, `detection_time` and `detected_changepoint` of the offline result, the number of final candidates `n_candidates`, and `statistics`, a data frame with one row per test statistic. The `[` method returns a `"focus_projections"` object and `as.matrix` an integer matrix.

Examples

```
set.seed(123)
Y <- c(rnorm(100, mean = 0), rnorm(100, mean = 2))

# Online interface
det <- detector_create(type = "univariate")
for (y in Y[1:120]) detector_update(det, y)
det
summary(det, family = "gaussian")
get_statistics(det, family = "gaussian")

# Offline interface
res <- focus_offline(Y, threshold = 20, type = "univariate",
                     family = "gaussian")
res
summary(res)
plot(res)
plot(res, data = Y)
if (requireNamespace("ggplot2", quietly = TRUE)) {
```

```
  ggplot2::autoplot(res, data = Y)
}

# Projection index sets
proj <- generate_projection_indexes(d = 6, p = 2)
proj
head(proj, 2)
as.matrix(proj)
```

`focus_offline` *Run a 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).

When more than one statistic is computed (e.g., the sum and maximum statistics of `"npfocus"`), a detection occurs when any statistic exceeds its threshold.

Value

An object of class `"focus_offline"`, with `print`, `summary` and `plot` methods (see `focus-methods`): 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")
result
summary(result)

# Plot the trace of the statistic, the threshold and the detection
plot(result)

# Plot the data above the trace of the statistic
plot(result, data = Y)

# 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

An object of class `"focus_projections"`: a list of integer vectors, each holding the 0-based column indices of one projection, with the number of dimensions and the projection size stored in the attributes `"d"` and `"p"`. It can be passed directly as the `dim_indexes` argument of `detector_create()` and `focus_offline()`. A `print` method shows one projection per row, subsetting with `[` (e.g., `head()`) keeps the class, and `as.matrix()` returns the indices as a matrix with one row per projection; see `focus-methods`.

Examples

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

# Use with multivariate detector
det <- detector_create(type = "multivariate", dim_indexes = proj)
detector_update(det, c(0.5, 1.2, -0.3, 0.8, 0.1))

# Offline comparison of the full detector and approximation
set.seed(42)
d <- 6

# 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(6, 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 the 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`	A `"focus_detector"` object 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 raises a warning and the parameter is ignored.

NPFOCuS. For non-parametric detection, the detector must be created with `detector_create(type = "npfocus", quantiles = ...)`. NPFOCuS returns two statistics (sum and maximum over the quantiles) as a vector; in the offline interface, `stat` is 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 and the (optional) pre-change mean `mu0_arp` are set at detector creation, so `theta0` is ignored (with a warning) for this family.

Value

An object of class "focus_statistics", with a print method (see focus-methods): a list with components

stopping_time	Numeric. Current time index (number of observations processed).
changepoint	Numeric or NULL. Estimated changepoint location, i.e., the index of the last observation before the change, or NULL if no estimate is available.
stat	Numeric scalar or vector, or NULL if no candidate is available yet. The test statistic: a scalar for all families except "npfocus", which returns the sum and the maximum of the statistics over the quantiles.

The family is stored in the attribute "family".

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

∗ **package**
    focus-package, 2
[.focus_projections *(*focus-methods*)*, 12

as.matrix.focus_projections
        *(*focus-methods*)*, 12
autoplot.focus_offline *(*focus-methods*)*,
        12

detector_candidates, *3*, 5
detector_cands_len, 6
detector_create, *3*, *5, 6*, 7, *10–12*, *16*, *18, 19*
detector_info_n, 9
detector_info_sn, 10, *14*
detector_update, *3*, *7, 8*, 11

focus *(*focus-package*)*, 2
focus-methods, 12
focus-package, 2
focus_offline, *3*, *12, 13*, 15, *18*

generate_projection_indexes, *3*, *8*, *12*, 17
get_statistics, *3*, *6–8*, *12, 13*, *16*, 19

matplot, *13*

plot.focus_offline *(*focus-methods*)*, 12
print.focus_detector *(*focus-methods*)*, 12
print.focus_offline *(*focus-methods*)*, 12
print.focus_projections
        *(*focus-methods*)*, 12
print.focus_statistics *(*focus-methods*)*,
        12
print.summary.focus_detector
        *(*focus-methods*)*, 12
print.summary.focus_offline
        *(*focus-methods*)*, 12

summary.focus_detector *(*focus-methods*)*,
        12
summary.focus_offline *(*focus-methods*)*,
        12