# Package 'csbewma'

September 10, 2026

**Title** Cumulative Standardized Binomial EWMA for Multiple Stream Processes

**Version** 1.1.0

**Author** Faruk Muritala [aut, cre],
Austin Brown [aut],
Dhrubajyoti Ghosh [aut],
Sherry Ni [aut]

**Maintainer** Faruk Muritala <fmurital@students.kennesaw.edu>

**Description** Implements the Cumulative Standardized Binomial Exponentially Weighted Moving Average (CSB-EWMA) control chart for monitoring multiple independent streams with binomial outcomes. Provides exact variance calculations, adaptive control limits, post-hoc identification with multiple testing corrections (Bonferroni, Holm, Benjamini-Hochberg), and visualization tools. The method is described in Muritala et al. (2026) <doi:10.48550/arXiv.2601.09968>.



**URL** https://github.com/fmurital/csbewma_R_package

**BugReports** https://github.com/fmurital/csbewma_R_package/issues

**Encoding** UTF-8

**Depends** R (>= 3.5)

**Imports** ggplot2, stats, patchwork, utils

**Suggests** testthat (>= 3.0.0)

**Config/roxygen2/version** 8.1.0

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2026-09-10 03:30:02 UTC

## Contents

---

`apply_multiple_testing` *Apply multiple testing corrections*

---

## Description

Applies Bonferroni, Holm, or Benjamini-Hochberg corrections.

## Usage

```
apply_multiple_testing(pvals, method = "BH", alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `pvals` | Numeric vector of raw p-values |
| `method` | Correction method: "BH", "bonferroni", "holm", or "raw" |
| `alpha` | Significance level (default = 0.05) |

## Value

List with adjusted_pvals and flags

## Examples

```
pvals <- c(0.001, 0.01, 0.03, 0.10, 0.50)
result <- apply_multiple_testing(pvals, method = "BH", alpha = 0.05)
```

## calculate_pvalues *Calculate p-values for all streams*

### Description

Computes exact binomial p-values for each stream based on successes.

### Usage

```
calculate_pvalues(bin_matrix, p0 = 0.5)
```

### Arguments

| | |
|---|---|
| bin_matrix | Matrix of binary indicators (streams as rows, time as columns) |
| p0 | In-control proportion (default = 0.5) |

### Value

Numeric vector of p-values

### Examples

```
bin_mat <- matrix(rbinom(500, 1, 0.5), nrow = 10, ncol = 50)
pvals <- calculate_pvalues(bin_mat)
```

## check_stream_correlation *Check Pairwise Correlation Among Streams for the Independence Assumption*

### Description

The CSB-EWMA control chart assumes that all monitored streams are independent (dissertation Section 6.2.7 and Limitations discussion). This function reports the pairwise Pearson correlation among a set of candidate streams and flags any pair whose absolute correlation exceeds a threshold, so the user can decide which streams to keep before calling `run_csb_ewma()` or `csb_ewma()`. It does not remove or adjust any stream automatically. No validated method for correcting correlated streams currently exists for this chart; see the companion paper's Discussion section, which lists correlated-stream handling as future work.

### Usage

```
check_stream_correlation(data, threshold = 0.3, use = "pairwise.complete.obs")
```

### Arguments

| | |
|---|---|
| data | A numeric matrix or data frame with candidate streams as columns. |
| threshold | Absolute Pearson correlation above which a pair is flagged. Default is 0.3, the threshold used in dissertation Section 6.2.7 to select independent streams for the Brisbane River application and restated generally in the Limitations discussion. This is a practical default carried over from that application, not a universally derived optimal value; adjust it if your own application calls for a different threshold. |
| use | Passed to `stats::cor()`. Default `"pairwise.complete.obs"`. |

### Value

A list with three elements: `correlation_matrix` (the full pairwise Pearson correlation matrix), `flagged_pairs` (a data frame of stream pairs with `abs(r) > threshold`, sorted by descending absolute correlation, zero rows if none), and `independence_ok` (`TRUE` if no pair exceeds the threshold).

### Examples

```
set.seed(1)
x <- matrix(rnorm(600), ncol = 6)
colnames(x) <- paste0("stream_", 1:6)
check_stream_correlation(x)
```

---

## csb_ewma *CSB-EWMA Control Chart*

---

### Description

Runs the Cumulative Standardized Binomial EWMA control chart on multiple stream data.

### Usage

```
csb_ewma(
  data,
  lambda,
  L,
  p0 = 0.5,
  max_time = NULL,
  stop_at_signal = TRUE,
  distribution = NULL,
  posthoc_method = "BH",
  alpha = 0.05,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | A matrix of binary indicators (0/1) with streams as rows and time as columns |
| `lambda` | Smoothing parameter for EWMA (0 < lambda <= 1) |
| `L` | Control limit multiplier |
| `p0` | In-control proportion (default = 0.5) |
| `max_time` | Maximum time points to monitor (default = NULL uses all) |
| `stop_at_signal` | Logical. If TRUE (default), monitoring stops at the first signal and post-hoc identification is performed once, stored in `flagged` (backward compatible with every released version through 1.1.0). If FALSE, monitoring continues through `max_time` and post-hoc identification is performed separately at each signal in `signal_times`, stored as a named list in `flagged_by_signal` (names are the signal times as character strings); `flagged` is then set to the first signal's result for convenience. |
| `distribution` | If data is continuous, specify distribution |
| `posthoc_method` | Method for post-hoc identification (default = "BH") |
| `alpha` | Significance level for post-hoc (default = 0.05) |
| `verbose` | Logical. If TRUE, prints informational messages about whether the input data required dichotomization. Default FALSE (silent). |

## Details

By default (`stop_at_signal = TRUE`), monitoring stops at the first signal and post-hoc identification is performed once, matching the behavior of every released version of this function through 1.1.0. Setting `stop_at_signal = FALSE` instead continues monitoring through `max_time` and performs post-hoc identification separately at every signal recorded in `signal_times`, using only the data observed up to and including that signal. See `run_csb_ewma` for the full explanation of continuation mode; it applies identically here since this function calls that one internally.

## Value

A list of class "csb_ewma" containing chart results and flagged streams. When `stop_at_signal = FALSE`, also contains `signal_times` (every time point at which a signal was detected) and `flagged_by_signal` (post-hoc identification results at each of those times).

## See Also

`run_csb_ewma()` for the lower-level engine this function calls, useful directly when a variance cache should be precomputed once and reused across many chart runs (for example, in simulation studies).

## Examples

```
set.seed(123)
bin_data <- matrix(rbinom(10*100, 1, 0.5), nrow = 10, ncol = 100)
for(i in 1:3) bin_data[i, ] <- rbinom(100, 1, 0.8)
result <- csb_ewma(bin_data, lambda = 0.175, L = 1.375)
```

```
print(result)
plot(result)

# Continue monitoring past the first signal and see every signal time
result2 <- csb_ewma(bin_data, lambda = 0.175, L = 1.375,
                    stop_at_signal = FALSE)
print(result2$signal_times)
```

---

`dichotomize_data` *Dichotomize Continuous Data to Binary Indicators*

---

## Description

Converts continuous observations to binary indicators based on whether each observation exceeds the in-control median or specified quantile.

## Usage

```
dichotomize_data(data, distribution, p0 = 0.5)
```

## Arguments

| | |
|---|---|
| `data` | Numeric vector of continuous observations |
| `distribution` | Character string specifying the distribution type |
| `p0` | In-control proportion (default = 0.5) |

## Value

Integer vector of binary indicators (0 or 1)

## Examples

```
x <- rnorm(100)
binary <- dichotomize_data(x, distribution = "normal", p0 = 0.5)
```

dichotomize_range *Convert Continuous Measurements to Binary Using an Acceptable Range*

## Description

Converts continuous measurements to binary indicators based on whether each value falls inside or outside an acceptable operating range. An observation is coded 1 if it falls below the lower bound or above the upper bound of the range, and 0 if it falls inside the range. This is the practical, real-data preprocessing rule used in dissertation Section 6.4.1 (Brisbane River water quality application), generalized here so the range bounds can come from either source a practitioner is likely to have.

## Usage

```
dichotomize_range(
  data,
  lower_limit = NULL,
  upper_limit = NULL,
  reference = NULL,
  lower_percentile = 0.01,
  upper_percentile = 0.99
)
```

## Arguments

| | |
|---|---|
| `data` | Numeric vector of continuous measurements to convert. |
| `lower_limit` | Known lower bound of the acceptable range. Provide together with `upper_limit`. Do not combine with `reference`. |
| `upper_limit` | Known upper bound of the acceptable range. Provide together with `lower_limit`. Do not combine with `reference`. |
| `reference` | Numeric vector of historical or Phase I reference data, used to estimate the range bounds when they are not already known. Do not combine with `lower_limit`/`upper_limit`. |
| `lower_percentile` | Empirical lower percentile used to estimate the range from `reference` (default 0.01). Used only when `reference` is supplied. |
| `upper_percentile` | Empirical upper percentile used to estimate the range from `reference` (default 0.99). Used only when `reference` is supplied. |

## Details

Known specification or standard limits: supply `lower_limit` and `upper_limit` directly. Use this when the acceptable range for a variable is already established, for example a regulatory or engineering specification, or values a domain expert already knows for their process.

Estimated from historical (Phase I) data: supply `reference` instead, along with `lower_percentile` and `upper_percentile` (default 0.01 and 0.99, the 1st and 99th percentile). The bounds are then

computed as the empirical percentiles of `reference`. This is the approach used in dissertation Section 6.4.1, where the acceptable range for each water quality variable was not independently known and was instead estimated from a Phase I reference period.

This is a separate function from `dichotomize_data()`, which generates and thresholds data from an assumed theoretical distribution for the simulation studies. The two are not interchangeable. This function never assumes a distribution, and `dichotomize_data()` is unchanged and still governs the simulation-study workflow.

## Value

Integer vector of binary indicators (0 or 1), the same length as `data`. A value of 1 means the observation fell outside the acceptable range.

## Examples

```
# Known specification limits (e.g. a regulatory or engineering standard)
measurements <- c(6.1, 7.0, 7.8, 9.0, 5.5)
dichotomize_range(measurements, lower_limit = 6.5, upper_limit = 8.5)

# Bounds estimated from historical data (dissertation Section 6.4.1 approach)
historical <- rnorm(1000, mean = 7, sd = 0.5)
new_readings <- c(6.8, 7.1, 9.2)
dichotomize_range(new_readings, reference = historical)
```

---

# dichotomize_range_multi

*Convert Multiple Variables to Binary Using Per-Variable Acceptable Ranges*

---

## Description

Applies `dichotomize_range()` separately to each column (variable) of a dataset, using a distinct range specification for each one. This is for the common practical case where different variables monitored together do not share one acceptable range. For example, an acceptable heart rate range differs by patient age group, and an acceptable height range differs by population subgroup; applying a single percentile or a single pair of limits uniformly across all variables would misrepresent variables that legitimately have different tolerances. This function does not choose or estimate a shared default range for variables that lack one; every variable's range must be specified explicitly through `specs`.

## Usage

```
dichotomize_range_multi(data, specs)
```

### Arguments

| | |
|---|---|
| data | A numeric matrix or data frame with variables (streams) as columns. Column names, if present, are used to match against the names of specs. |
| specs | A named list, one element per column of data. Every column of data must have a matching entry in specs (by column name); an unspecified variable is treated as an error rather than silently defaulted, so that each variable's acceptable range is always a deliberate, documented choice. Each entry is itself a list of named arguments passed to dichotomize_range(): either list(lower_limit = , upper_limit = ) or list(reference = ,lower_percentile = , upper_percentile = ). |

### Details

Each element of specs is itself a list of arguments passed directly to dichotomize_range() for that column, so it follows the same rules: supply either lower_limit and upper_limit (known specification or standard limits for that variable), or reference (optionally with lower_percentile and upper_percentile) to estimate the range for that variable from its own historical data. Different variables may use different approaches; nothing requires them to match.

This function builds on dichotomize_range() and does not modify it. It is a dataset-level convenience wrapper only. dichotomize_data() (the simulation-study distributional method) remains entirely separate and is unaffected by this function.

### Value

An integer matrix with the same dimensions and column names as data (coerced to a data frame internally if a matrix was supplied), where each column is the binary result of applying that column's own range specification via dichotomize_range().

### Examples

```
# Two variables with genuinely different, manually specified ranges,
# e.g. distinct acceptable bands for two water quality measurements
water_data <- data.frame(
  pH = c(6.9, 7.2, 8.6, 5.9, 7.0),
  turbidity = c(2.1, 15.4, 3.0, 4.2, 50.0)
)
specs <- list(
  pH = list(lower_limit = 6.5, upper_limit = 8.5),
  turbidity = list(lower_limit = 0, upper_limit = 10)
)
dichotomize_range_multi(water_data, specs)
```

---

## flagged_streams_summary

*Summarize flagged streams*

---

## Description

Creates a summary of which streams were flagged.

## Usage

```
flagged_streams_summary(flagged_results)
```

## Arguments

| | |
|---|---|
| flagged_results | Output from identify_ooc() |

## Value

List with flagged streams and summary table

## Examples

```
bin_mat <- matrix(rbinom(10*100, 1, 0.5), nrow = 10, ncol = 100)
for(i in 1:3) bin_mat[i, ] <- rbinom(100, 1, 0.8)
result <- identify_ooc(bin_mat)
summary <- flagged_streams_summary(result)
print(summary$flagged_streams)
```

---

# generate_continuous_data

*Generate Continuous Data from Specified Distribution*

---

## Description

Generates continuous observations from normal, Laplace, uniform, or exponential distributions with optional shift parameter for out-of-control simulation.

## Usage

```
generate_continuous_data(distribution, n, shift = 0, p0 = 0.5)
```

## Arguments

| | |
|---|---|
| distribution | Character string: "normal", "laplace", "uniform", or "exponential" |
| n | Number of observations to generate |
| shift | Amount to shift the proportion (default = 0) |
| p0 | In-control proportion (default = 0.5) |

## Value

A numeric vector of length n containing the generated data

### Examples

```
data <- generate_continuous_data("normal", n = 100, shift = 0.2)
```

---

## identify_ooc *Identify out-of-control streams*

---

### Description

Main function for post-hoc identification using multiple testing corrections.

### Usage

```
identify_ooc(bin_matrix, alpha = 0.05, method = "BH", p0 = 0.5)
```

### Arguments

| | |
|---|---|
| bin_matrix | Matrix of binary indicators (streams as rows, time as columns) |
| alpha | Significance level (default = 0.05) |
| method | Correction method (default = "BH") |
| p0 | In-control proportion (default = 0.5) |

### Value

Data frame with p-values and flags for each stream

### Examples

```
set.seed(123)
bin_mat <- matrix(rbinom(10*100, 1, 0.5), nrow = 10, ncol = 100)
for(i in 1:3) bin_mat[i, ] <- rbinom(100, 1, 0.8)
result <- identify_ooc(bin_mat)
print(result[result$flagged, ])
```

---

## plot.csb_ewma *Plot CSB-EWMA Control Chart*

---

### Description

Creates a professional control chart showing EWMA statistic and limits.

### Usage

```
## S3 method for class 'csb_ewma'
plot(x, title = "CSB-EWMA Control Chart", show_signal = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | csb_ewma object from csb_ewma() function |
| title | Plot title (default = "CSB-EWMA Control Chart") |
| show_signal | Whether to highlight signal point (default = TRUE) |
| ... | Additional arguments passed to ggplot |

### Value

A ggplot object (invisibly) and displays the plot

### Examples

```
# See csb_ewma() for examples
```

---

## plot_chart_with_flagged *Combined Diagnostic Dashboard*

---

### Description

Creates a combined plot showing both the CSB-EWMA control chart and the flagged streams bar plot side by side or stacked.

### Usage

```
plot_chart_with_flagged(chart_result, flagged_results, layout = "side")
```

### Arguments

| | |
|---|---|
| chart_result | csb_ewma object from csb_ewma() function |
| flagged_results | Output from identify_ooc() function |
| layout | Either "side" for side-by-side or "stacked" for vertical |

### Value

A combined ggplot object (invisibly) and displays the plot

## plot_csb_ewma_direct *Direct Plot for CSB-EWMA Results*

### Description

Creates a professional control chart showing EWMA statistic and limits. This function can be called directly without S3 dispatch.

### Usage

```
plot_csb_ewma_direct(
  result,
  title = "CSB-EWMA Control Chart",
  show_signal = TRUE
)
```

### Arguments

| | |
|---|---|
| result | csb_ewma object from csb_ewma() or run_csb_ewma() |
| title | Plot title (default = "CSB-EWMA Control Chart") |
| show_signal | Whether to highlight signal point (default = TRUE) |

### Value

A ggplot object (invisibly) and displays the plot

## plot_flagged_streams *Plot Flagged Streams Bar Chart*

### Description

Creates a bar plot showing -log10(p-values) for each stream. Flagged streams appear in red, others in gray.

### Usage

```
plot_flagged_streams(flagged_results, alpha = 0.05, title = "Stream P-values")
```

### Arguments

| | |
|---|---|
| flagged_results | Output data frame from identify_ooc() function |
| alpha | Significance level for reference line (default = 0.05) |
| title | Plot title (default = "Stream P-values") |

## Value

A ggplot object (invisibly) and displays the plot

---

`precompute_variance` *Precompute variance vector for all time points*

---

## Description

This function precomputes the exact variance for all time points from 1 to max_t. For efficiency, it computes exact variances up to a convergence threshold (converge_t) and then sets remaining values to 1 (asymptotic). Based on the derivation, variance reaches 99% of asymptotic value by t=227, so converge_t = 500 is safe and efficient.

## Usage

```
precompute_variance(lambda, max_t, converge_t = 500)
```

## Arguments

| | |
|---|---|
| `lambda` | Smoothing parameter for EWMA |
| `max_t` | Maximum time point to precompute variance for |
| `converge_t` | Time point after which variance is set to 1 (asymptotic) |

## Value

A numeric vector of length max_t containing variance at each time point

## Examples

```
# Precompute variance for lambda = 0.175, up to t = 1000
var_cache <- precompute_variance(lambda = 0.175, max_t = 1000, converge_t = 500)
```

---

`print.csb_ewma` *Prints a formatted summary of CSB-EWMA chart results.*

---

## Description

Prints a formatted summary of CSB-EWMA chart results.

## Usage

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

### Arguments

| | |
|---|---|
| x | csb_ewma object from csb_ewma() or run_csb_ewma() function |
| ... | Additional arguments (not used) |

### Value

Invisibly returns the object

---

## rlaplace *Generate Laplace (Double Exponential) Random Variables*

---

### Description

Generates random numbers from the Laplace distribution using inverse transform sampling.

### Usage

```
rlaplace(n, location = 0, scale = 1)
```

### Arguments

| | |
|---|---|
| n | Number of observations to generate |
| location | Location parameter (median) of the distribution (default = 0) |
| scale | Scale parameter (spread) of the distribution (default = 1) |

### Value

A numeric vector of length n containing Laplace random variables

### Examples

```
## Not run:
x <- rlaplace(100, location = 0, scale = 1)

## End(Not run)
```

`run_csb_ewma` *Run CSB-EWMA Chart on Binary Data*

## Description

This function implements the core CSB-EWMA monitoring algorithm exactly as implemented in the original simulation code.

## Usage

```
run_csb_ewma(
  bin_matrix,
  lambda,
  L,
  var_cache,
  max_time = NULL,
  p0 = 0.5,
  stop_at_signal = TRUE
)
```

## Arguments

| | |
|---|---|
| `bin_matrix` | A matrix of binary indicators (streams as rows, time as columns) |
| `lambda` | Smoothing parameter for EWMA (0 < lambda <= 1) |
| `L` | Control limit multiplier |
| `var_cache` | Precomputed variance vector from precompute_variance() |
| `max_time` | Maximum time points to monitor (default = NULL uses all) |
| `p0` | In-control proportion (default = 0.5) |
| `stop_at_signal` | Logical. If TRUE (default), monitoring stops at the first signal, reproducing the behavior of every version through 1.1.0. If FALSE, monitoring continues through max_time and every signal time is recorded in signal_times. |

## Details

The algorithm works as follows:

1. Initialize cumulative sum (cum_sum = 0) and EWMA (r_prev = 0)
2. For each time point t = 1, 2, ..., max_time:
   - Get binary vector for current time point
   - Calculate C_t = sum of binary indicators
   - Update cumulative sum: cum_sum = cum_sum + C_t
   - Compute standardized statistic: W_t = (cum_sum - mu0*t) / sqrt(t*sigma2_0)
   - Update EWMA: r_t = lambda * W_t + (1 - lambda) * r_prev
   - Get exact variance from precomputed cache: v_t = var_cachet

- Compute control limits: UCL_t = L * sqrt(v_t), LCL_t = -L * sqrt(v_t)
- If r_t > UCL_t or r_t < LCL_t, a signal is recorded at time t
- Update r_prev = r_t and continue, unless stop_at_signal ends monitoring here

By default (`stop_at_signal = TRUE`), monitoring stops at the first signal, matching the behavior of every released version of this function through 1.1.0. Setting `stop_at_signal = FALSE` instead continues monitoring through `max_time`, recording every time point at which a signal condition is met in `signal_times`, without resetting the cumulative sum or the EWMA statistic after a signal; the formula above is unchanged, only the decision to stop early changes. This continuation mode is a practical convenience for real-time or dashboard-style monitoring where the process keeps running after an alarm; it is not itself a procedure described in the dissertation, which presents only the stop-at-first-signal convention.

## Value

A list of class "csb_ewma" containing chart results. Includes signal_time and signal_detected (the first signal, kept for backward compatibility) and signal_times (an integer vector of every time point at which a signal was recorded; empty if none).

## See Also

`csb_ewma()` for the higher-level convenience wrapper that dichotomizes continuous data, precomputes variance, calls this function, and performs post-hoc identification automatically.

## Examples

```
bin_matrix <- matrix(rbinom(10*200, 1, 0.5), nrow = 10, ncol = 200)
for(i in 1:3) bin_matrix[i, ] <- rbinom(100, 1, 0.8)
var_cache <- precompute_variance(0.175, max_t = 200)
result <- run_csb_ewma(bin_matrix, lambda = 0.175, L = 1.375, var_cache)
print(paste("Signal at time:", result$signal_time))

# Continue monitoring past the first signal
result2 <- run_csb_ewma(bin_matrix, lambda = 0.175, L = 1.375, var_cache,
                        stop_at_signal = FALSE)
print(result2$signal_times)
```

---

# `var_rt_exact_single` *Compute exact CSB-EWMA variance for a single time point*

---

## Description

This function implements the exact variance formula derived in Theorem 2. The variance is computed using double summation over the covariance structure of the standardized statistics. For a given smoothing parameter lambda and time point t, this returns Var(r_t) as defined in Equation (19) of the supplementary material.

## Usage

```
var_rt_exact_single(lambda, t)
```

## Arguments

| | |
|---|---|
| lambda | Smoothing parameter for EWMA. Must be between 0 and 1. Typical values range from 0.05 to 0.5. |
| t | Time point (positive integer). The variance is computed for this specific time point. |

## Value

The exact variance Var(r_t) at time t. For t = 0, returns 0.

## Examples

```
# Compute variance at time 10 for lambda = 0.175
var_rt_exact_single(lambda = 0.175, t = 10)

# Compute variance at time 50 for lambda = 0.15
var_rt_exact_single(lambda = 0.15, t = 50)
```

# Index