Package: csbewma (via r-universe)

June 11, 2026

Title Cumulative Standardized Binomial EWMA for Multiple Stream Processes

Version 1.0.1

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>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5)

Imports ggplot2, stats, patchwork

Suggests testthat (>= 3.0.0)

NeedsCompilation no

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

Date/Publication 2026-06-05 14:50:02 UTC

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

RemoteRef HEAD

RemoteSha 3fb8ee5719ca29fbe4264c2d6d909fb00e895694

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(s). Either a single number or a vector of length = nrow(bin_matrix)

Value

Numeric vector of p-values

Examples

```
bin_mat <- matrix(rbinom(500, 1, 0.5), nrow = 10, ncol = 50)
pvals <- calculate_pvalues(bin_mat, p0 = 0.5)
```

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,
  posthoc_method = "BH",
  alpha = 0.05,
  verbose = TRUE
)
```

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(s). Either a single number (same for all streams) or a numeric vector of length equal to number of rows in data.
max_time	Maximum time points to monitor (default = NULL uses all)
posthoc_method	Method for post-hoc identification (default = "BH")
alpha	Significance level for post-hoc (default = 0.05)
verbose	If TRUE, print informational messages (default = TRUE)

Value

A list of class "csb_ewma" containing chart results and flagged streams

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)
message(result)
plot(result)
}
```

---

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)
```

`flagged_streams_summary` *Summarize flagged streams*

Description

Creates a summary of which streams were flagged.

Usage

```
flagged_streams_summary(flagged_results, verbose = TRUE)
```

Arguments

`flagged_results`	Output from identify_ooc()
`verbose`	If TRUE, print informational messages (default = TRUE)

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

Description

Identify out-of-control streams Main function for post-hoc identification using multiple testing corrections.

Usage

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

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(s). Either a single number or a vector of length = nrow(bin_matrix)
verbose	If TRUE, print informational messages (default = TRUE)

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, p0 = 0.5)
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

```
{
x <- rlaplace(100, location = 0, scale = 1)
}
```

`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)
```

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(s). Either a single number (same for all streams) or a numeric vector of length equal to number of rows in bin_matrix. Example p0 = c(0.5, 0.6, 0.7) # different p0 for stream1, stream2, stream3.

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, signal at time t and break
   - Otherwise, update r_prev = r_t and continue

Value

A list of class "csb_ewma" containing chart results

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)
message(paste("Signal at time:", result$signal_time))
```

---

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