

Package ‘driftwatch’

October 7, 2026

Title Sequential Monitoring of Item Parameter Drift

Version 0.1.0

Description Ongoing surveillance of item parameter drift for continuous testing programs and pre-equated item banks. Estimates item difficulty in rolling calibration windows, runs sequential cumulative sum (CUSUM) detection (Page, 1954, <[doi:10.1093/biomet/41.1-2.100](https://doi.org/10.1093/biomet/41.1-2.100)>; applied to testing by Veerkamp and Glas, 2000, <[doi:10.3102/10769986025004373](https://doi.org/10.3102/10769986025004373)>) with change-point estimation, separates gradual drift from abrupt jumps, tunes alarm thresholds for a bank-wide false-alarm target by simulation on the program's own design, quantifies score and pass-rate impact, and records recommended actions in an audit log.

License MIT + file LICENSE

URL <https://github.com/edidatasolutions/driftwatch>,
<https://edidatasolutions.github.io/driftwatch/>

BugReports <https://github.com/edidatasolutions/driftwatch/issues>

Encoding UTF-8

Depends R (>= 4.1)

Imports stats, utils

Suggests knitr, markdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Daniel Edi [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0001-5475-819X>>)

Maintainer Daniel Edi <danieledi2026@gmail.com>

Repository CRAN

Date/Publication 2026-10-07 09:50:16 UTC

Contents

dw_actions	2
dw_estimate	3
dw_impact	4
dw_monitor	5
dw_simulate	6
dw_tune	7
dw_twopoint	8
Index	9

dw_actions	<i>Recommended actions and audit log</i>
------------	--

Description

- Turns monitoring alarms into recommended actions with a written rationale:
- abrupt change of at least ‘retire_at’ logits: ‘retire’ (consistent with exposure/compromise or a key or rendering error; review the key, exposure counts and any content change before reuse);
 - other abrupt changes, gradual drift and undetermined changes: ‘recalibrate’ (undetermined ones should be reclassified later);
 - any alarm on an anchor item: additionally ‘remove from anchor set’.

Usage

```
dw_actions(  
  monitor,  
  anchors = character(),  
  retire_at = 0.5,  
  analyst = NA_character_  
)
```

Arguments

monitor	A ‘dw_monitor’.
anchors	Item ids used as equating anchors.
retire_at	Abrupt magnitude (logits) that triggers retirement.
analyst	Name recorded in the log.

Value

Audit-log data frame, one row per alarmed item.

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
mon <- dw_monitor(dw_estimate(sim$responses, sim$bank), h = 8)
log <- dw_actions(mon, anchors = sim$bank$item[1:10], analyst = "DE")
if (nrow(log)) log[, c("item", "type", "magnitude", "action")]
```

dw_estimate	<i>Window-level item difficulty estimates</i>
-------------	---

Description

Rasch difficulty for every item x window, with examinee ability treated as known (from operational scoring on the rest of the form). Estimation is penalized maximum likelihood with a weak $N(b_bank, 'prior_sd'^2)$ penalty that only matters for all-correct or all-incorrect windows. Newton steps run for all item-windows at once.

Usage

```
dw_estimate(responses, bank, prior_sd = 3)
```

Arguments

responses	Long data frame: 'window' (integer), 'item', 'theta', 'x'.
bank	Reference parameters: 'item', 'b', and optionally 'se'.
prior_sd	SD of the weak penalty.

Value

A 'dw_estimates' object: matrices 'b_hat', 'se', 'n' and 'z' (items x windows; 'z' is the standardized deviation from the bank, NA where an item was not administered), plus 'bank' and 'responses'.

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
est <- dw_estimate(sim$responses, sim$bank)
round(est$z[1:5, 1:6], 2)
```

dw_impact

*Score and pass-rate impact of drifted items***Description**

For a pre-equated form scored by true-score conversion (the raw score is mapped to theta through the test characteristic curve of the scoring parameters), an examinee of ability theta is reported at $\text{TCC_scoring}^{-1}(\text{TCC_current}(\theta))$. The function compares scoring scenarios against the current (true or best-estimate) difficulties:

keep Score with banked parameters for every item.

remove Drop flagged items from the form; score the rest with banked parameters.

recalibrate Score flagged items with their current estimates.

and reports reported-score bias at the cut and the pass rate in the population against the correct pass rate.

Usage

```
dw_impact(
  form,
  bank,
  current,
  flagged,
  recalibrated = NULL,
  cut = 0,
  theta_mean = 0,
  theta_sd = 1
)
```

Arguments

form	Item ids on the form.
bank	Banked parameters ('item', 'b').
current	Named vector of current difficulties: the truth in a simulation, or the latest window estimates in practice.
flagged	Item ids flagged by monitoring.
recalibrated	Named vector of re-estimated difficulties for flagged items (default: 'current[flagged]').
cut	Passing standard on the theta scale.
theta_mean, theta_sd	Examinee population.

Value

Data frame: 'scenario', 'n_items', 'bias_at_cut', 'mean_abs_bias', 'pass_rate', 'pass_rate_error' (vs the correct rate).

Examples

```
# A 20-item form where one item became 0.8 logits harder
current <- setNames(c(0.8, rep(0, 19)), paste0("q", 1:20))
bank <- data.frame(item = names(current), b = 0)
dw_impact(names(current), bank, current, flagged = "q1", cut = 0)
```

dw_monitor

Sequential drift monitoring

Description

Runs a two-sided CUSUM on each item's standardized deviations $z_t = (b_t - b_{\text{bank}}) / \text{SE}$: $S^+[t] = \max(0, S^+[t-1] + z[t] - k)$ and likewise downward. An alarm is raised the first window either statistic exceeds h . Drift type is then classified by comparing step and ramp fits to the difficulty series (profiling the change point), using data up to `followup` windows after the alarm. `change_point` is the onset under the better-fitting model; `cusum_change_point` is the standard CUSUM estimator (the window after the statistic last left zero), which locates abrupt changes well but lags gradual onsets.

Usage

```
dw_monitor(estimates, h, k = 0.5, followup = 3, min_log_lr = 1)
```

Arguments

<code>estimates</code>	A <code>'dw_estimates'</code> object.
<code>h</code>	Alarm threshold, ideally from <code>[dw_tune()]</code> .
<code>k</code>	Reference value (half the shift, in SE units, the chart is tuned to detect).
<code>followup</code>	Windows after the alarm used to classify the change (<code>'Inf'</code> = all data to date).
<code>min_log_lr</code>	Minimum log likelihood ratio between the step and ramp models for a type to be assigned; weaker evidence gives <code>"undetermined"</code> . Soon after an alarm a ramp and a step often cannot be told apart; rerun with more windows to resolve them.

Value

A `'dw_monitor'` object; `$items` has one row per item: `'alarm'` (logical), `'alarm_window'`, `'direction'` (`'harder'` / `'easier'`), `'change_point'`, `'type'` (`'abrupt'` / `'gradual'` / `'undetermined'`), `'type_log_lr'` (evidence for the better model), `'magnitude'` (current displacement under the better model, logits), `'rate'` (logits per window, gradual only).

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
est <- dw_estimate(sim$responses, sim$bank)
mon <- dw_monitor(est, h = 8)
mon
table(alarm = mon$items$alarm, truth = sim$truth$type)
```

dw_simulate

*Simulate a continuously administered item bank with known drift***Description**

Each window, every item is answered by a Poisson number of examinees whose abilities are known from operational scoring (the population mean may trend over time; this is not drift). Items are stable, drift gradually (linear from an onset window) or jump abruptly at an onset window.

Usage

```
dw_simulate(
  n_items = 300,
  n_windows = 40,
  mean_n = 80,
  p_gradual = 0.1,
  p_abrupt = 0.05,
  slope_range = c(0.02, 0.06),
  jump_range = c(0.4, 1),
  onset_range = c(5, 30),
  theta_trend = 0.01,
  ref_se = 0.05,
  seed = NULL
)
```

Arguments

n_items, n_windows	Bank size and number of windows.
mean_n	Mean responses per item per window.
p_gradual, p_abrupt	Share of items with each drift type.
slope_range	Absolute gradual slope per window (logits).
jump_range	Absolute abrupt jump (logits).
onset_range	Windows in which drift can begin.
theta_trend	Change in examinee mean ability per window.
ref_se	Standard error of the banked (reference) difficulties.
seed	Optional seed.

Value

A 'dw_sim': '\$responses' ('window', 'item', 'theta', 'x'), '\$bank' ('item', 'b', 'se'), '\$truth' ('item', 'type', 'onset', 'size', 'b_true_final') and '\$b_path' (items x windows matrix of true difficulty).

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
table(sim$truth$type)
```

dw_tune

Tune the alarm threshold for a bank-wide false-alarm target

Description

An item raises a false alarm over the monitoring horizon exactly when the maximum of its CUSUM statistics exceeds ‘h’. So ‘h’ is the ‘1 - target’ quantile of that maximum under no drift. With ‘method = "design"’, the null is simulated on the program’s own design: the same items, windows, examinee abilities and sample sizes, with responses regenerated from the banked difficulties, then re-estimated and re-standardized exactly as in monitoring. This captures small-sample non-normality of ‘z’ and sparse windows. ‘method = "normal"’ treats ‘z’ as iid $N(0, 1)$, which is fast and useful for planning a bank that does not exist yet.

Usage

```
dw_tune(
  estimates = NULL,
  target = 0.01,
  k = 0.5,
  method = c("design", "normal"),
  n_rep = 20,
  n_windows = NULL,
  seed = NULL
)
```

Arguments

estimates	A ‘dw_estimates’ object (required for “design”).
target	Probability that a non-drifting item alarms at least once over the horizon. Expected false alarms for the bank = ‘target * n_items’.
k	Reference value (as in [dw_monitor()]).
method	“design” or “normal”.
n_rep	Null replicates of the whole bank (“design”) or simulated item series (“normal”).
n_windows	Horizon for “normal” (default: the estimates’ windows).
seed	Optional seed.

Value

A list: ‘h’, ‘target’, ‘k’, ‘method’, ‘expected_false_alarms’ (per bank, when estimates are given), and ‘null_max’ (the simulated maxima).

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
est <- dw_estimate(sim$responses, sim$bank)
dw_tune(est, target = 0.02, method = "normal", seed = 1)$h

# Design-based tuning (recommended) simulates the whole bank n_rep times.
dw_tune(est, target = 0.02, n_rep = 5, seed = 1)$h
```

dw_twopoint	<i>Two-point drift check (the conventional baseline)</i>
-------------	--

Description

Pools the first and last blocks of windows, estimates each item's difficulty in both, and flags items whose robust z of the difference (median/MAD standardized) exceeds 'crit': the usual displacement check at equating time.

Usage

```
dw_twopoint(estimates, early = 1:5, late = NULL, crit = 2.7)
```

Arguments

estimates	A 'dw_estimates' object.
early, late	Window indices forming the two calibrations.
crit	Robust-z criterion.

Value

Data frame: 'item', 'd', 'robust_z', 'flag'.

Examples

```
sim <- dw_simulate(n_items = 60, n_windows = 20, mean_n = 60,
                  onset_range = c(5, 12), seed = 1)
tp <- dw_twopoint(dw_estimate(sim$responses, sim$bank))
table(flag = tp$flag, truth = sim$truth$type)
```

Index

`dw_actions`, [2](#)
`dw_estimate`, [3](#)
`dw_impact`, [4](#)
`dw_monitor`, [5](#)
`dw_simulate`, [6](#)
`dw_tune`, [7](#)
`dw_twopoint`, [8](#)