# When Your Model Stops Working: Anytime-Valid Calibration Monitoring

Tristan Farran
*MSc Computational Science, University of Amsterdam*
tristan.farran@student.uva.nl

February 26, 2026

**Abstract**

Deployed machine learning models often experience calibration drift as the world changes. To address this challenge, we present PITMonitor, a sequential method for detecting calibration changes in probabilistic models. Unlike residual-based drift detectors, PITMonitor operates directly on probability integral transforms, making it sensitive to distributional changes — systematic over- or under-confidence, shifts in predictive mean or variance — that can leave accuracy metrics unchanged. Unlike many traditional calibration tests, PITMonitor provides *anytime-valid* false alarm control: the probability of ever raising a spurious alarm is bounded by $\alpha$ for all time, without requiring a fixed horizon or stopping rule. We prove Type I error control via Ville's inequality and demonstrate detection power on three scenarios from `river`'s FriedmanDrift dataset, comparing performance against the seven included stream-drift detectors. Code is available at `https://github.com/tristan-farran/pitmon`.

## 1 Introduction

Probabilistic models deployed in production face a fundamental challenge: the world changes. Across many essential domains from medicine to finance, models may encounter non-stationary processes, regime shifts, and concept drift. When these shifts occur, model calibration can degrade drastically, leading to consequential issues downstream. In practice, calibration is often monitored using ad-hoc procedures such as periodic recalibration schedules, rolling-window hypothesis tests, threshold-based alerts on summary metrics, or manual inspection of residuals.

These approaches suffer from a fundamental statistical problem: *they do not control the false alarm rate over continuous monitoring.* A practitioner who checks calibration daily with a $p < 0.05$ threshold will, over a year of monitoring, almost certainly observe spurious alarms even if the model remains stable. Classical hypothesis tests assume a fixed sample size determined before seeing data, continuous monitoring violates this assumption.

More principled alternatives are provided by online drift detectors, such as those implemented in the `river` library (**?**). Classical detectors including DDM, EDDM, and KSWIN are lightweight, easy to deploy, and effective at detecting abrupt changes, but are typically based on heuristic thresholds or fixed-sample statistical arguments and do not provide explicit long-run false alarm guarantees under continuous monitoring or changepoint localisation.

ADWIN employs adaptive windowing with statistical change detection via Hoeffding bounds (**?**). Its parameter $\delta$ bounds the false alarm probability *per comparison at each time step* (with a Bonferroni-style correction $\delta' = \delta/n$ for window length $n$), not the probability of ever raising a false alarm over an unbounded stream. Because ADWIN evaluates its Hoeffding bound at every incoming observation, the number of comparisons grows with the monitoring horizon: the stream-level false alarm probability therefore increases with deployment duration, even when the per-step bound is held constant. Aligning $\delta$ with a desired stream-level FPR requires knowing the monitoring window length in advance and deriving or calibrating the relationship — neither of which is required for PITMonitor. Furthermore, ADWIN operates on generic accuracy signals such as squared residuals, conflating calibration drift with accuracy degradation: a model that becomes systematically overconfident — predicting uncertainty intervals that are too narrow — while retaining the same point-prediction accuracy would change the PIT distribution but leave the squared residual distribution largely unchanged. PITMonitor, operating directly on PIT values, remains sensitive to exactly this class of calibration failure. ADWIN also does not provide changepoint estimates, only partially addressing the problem of reliable, informative, long-term calibration monitoring.

We propose PITMonitor, an anytime-valid calibration monitoring method with four key properties:

1. **Anytime-valid false alarm control:** we prove that $\mathbb{P}(\text{ever alarm} \mid H_0) \leq \alpha$ for all time, without requiring a pre-specified monitoring horizon or stopping rule.
2. **Change detection and localisation without static-error alarms:** PITMonitor detects and locates *changes* in the PIT process. A model that is consistently miscalibrated but stable will not trigger alarms, while a changing process can.[1]
3. **No baseline period required:** unlike methods requiring a "clean" reference distribution, PITMonitor works from the first observation by testing the PIT sequence's exchangeability.
4. **Practical efficiency:** the exact algorithm runs in $O(t \log t)$ time and $O(t)$ space for $t$ observations, with a simple recursive update.

## 2 Background

### 2.1 Probability Integral Transforms

A probabilistic model outputting a predicted cumulative distribution function $\hat{F}$ over outcomes is *calibrated* if these predictions match reality: among all predictions where $\hat{F}(y) = p$, the outcome $Y \leq y$ should occur roughly $(100 \times p)\%$ of the time.

The *probability integral transform* (PIT) provides a universal tool for assessing calibration (**?**). For a continuous predictive CDF $F$ and realized outcome $y$, the PIT is $U = F(y)$. A classical result states that if $F$ is the true distribution of $Y$, then $U \sim \text{Uniform}(0, 1)$.

In the regression setting where the model outputs a Gaussian predictive distribution $\mathcal{N}(\mu_t, \sigma_t^2)$, the PIT is:

$$U_t = \Phi\left(\frac{y_t - \mu_t}{\sigma_t}\right) \tag{1}$$

where $\Phi$ denotes the standard normal CDF. Under perfect calibration this gives $U_t \sim \text{Uniform}(0, 1)$.

[1] In many domains some amount of miscalibration is inevitable, but model degradation remains a pressing concern.

For discrete outcomes (e.g., classification), randomization yields a continuous PIT (**?**). Given predicted class probabilities $(\hat{p}_1, \ldots, \hat{p}_K)$ and true class $y \in \{1, \ldots, K\}$:

$$U = \sum_{j=1}^{y-1} \hat{p}_j + V \cdot \hat{p}_y, \quad V \sim \text{Uniform}(0, 1) \tag{2}$$

placing $U$ uniformly within the cumulative probability interval corresponding to the true class.

### 2.2 Exchangeability

A sequence $(X_1, X_2, \ldots)$ is *exchangeable* if its joint distribution is invariant to finite permutations. Exchangeability is weaker than independence: i.i.d. sequences are exchangeable, but exchangeable sequences need not be independent (**?**).

**Remark 1** (Stable Miscalibration Preserves Exchangeability)**.** *If a model is consistently miscalibrated, with its calibration error distribution remaining stable over time, the resulting PITs are i.i.d. from some fixed, non-uniform distribution. Since i.i.d. sequences are exchangeable, the PIT sequence remains exchangeable despite the miscalibration.*

This observation is central to PITMonitor's design:

- **Perfect calibration:** PITs are i.i.d. Uniform$(0, 1) \Rightarrow$ exchangeable
- **Stable miscalibration:** PITs are i.i.d. from a non-uniform distribution $\Rightarrow$ still exchangeable
- **PIT-process change:** the PIT distribution changes at some time $\tau \Rightarrow$ not exchangeable

By testing exchangeability rather than uniformity, we avoid triggering on stable calibration error.

It should be noted, however, that non-exchangeability can also result from temporal dependence: autocorrelated PITs can trigger alarms even when the calibration distribution is unchanged. This can occur in time series models, models with lagged features, or whenever predictions are not independent across time. Practitioners should check for autocorrelation in the PIT sequence and be cautious when interpreting alarms in settings where temporal dependence is expected.

### 2.3 Conformal P-values

To sequentially test exchangeability we employ *conformal p-values from ranks* (**?**).

Given observations $U_1, \ldots, U_t$, define the rank of $U_t$:

$$R_t = \#\{s \leq t : U_s \leq U_t\} \tag{3}$$

**Proposition 1** (Rank Uniformity under Exchangeability)**.** *If $(U_1, \ldots, U_t)$ is exchangeable, then the rank $R_t$ is uniformly distributed on $\{1, \ldots, t\}$.*

*Proof.* By exchangeability, $(U_1, \ldots, U_t)$ is equally likely to be in any of the $t!$ orderings. For any fixed rank $r \in \{1, \ldots, t\}$, exactly $(t-1)!$ of these orderings place $U_t$ in position $r$. Therefore $\mathbb{P}(R_t = r) = (t-1)!/t! = 1/t$, giving uniform distribution on $\{1, \ldots, t\}$. □

To obtain continuous uniform p-values from the discrete uniform ranks, we randomize within ties:

$$p_t = \frac{R_t - 1 + V_t}{t}, \quad V_t \sim \text{Uniform}(0, 1) \tag{4}$$

Under $H_0$ (exchangeability), these p-values are marginally Uniform$(0, 1)$. After a changepoint, exchangeability breaks: new PITs come from a shifted mechanism and systematically rank higher or lower than pre-change PITs. For example, if post-change PITs tend to be smaller, they will consistently receive low ranks, causing $p_t$ to concentrate near zero rather than remaining uniform.

### 2.4 E-values

An *e-value* is a non-negative random variable $E$ satisfying $\mathbb{E}[E] \leq 1$ under the null hypothesis (**?**). By Markov's inequality, $\mathbb{P}(E \geq 1/\alpha) \leq \alpha$, so thresholding at $1/\alpha$ yields a valid level-$\alpha$ test without needing to know the distribution of $E$ under the null. This is in contrast to p-values, which require knowing the null distribution explicitly to calibrate thresholds.

Under alternatives where the null is violated, an e-value has power if $\mathbb{E}[E] > 1$. The density-based construction in Section **??** achieves this adaptively: when conformal p-values concentrate in certain bins due to non-exchangeability, the histogram places more mass there, yielding $\mathbb{E}[e] > 1$ without requiring a parametric specification of the alternative.

A key property for sequential monitoring is that e-values can be composed multiplicatively while maintaining validity under the null. If $E_1, E_2$ are conditional e-values with $\mathbb{E}[E_1 \mid \mathcal{F}_0] \leq 1$ and $\mathbb{E}[E_2 \mid \mathcal{F}_1] \leq 1$ (where $\mathcal{F}_t$ is the information available through time $t$), their product remains a valid e-value. This allows us to define a cumulative e-process:

$$M_t = E_1 \times \cdots \times E_t, \quad M_t = M_{t-1} \cdot E_t \tag{5}$$

Taking conditional expectations given past observations:

$$\mathbb{E}[M_t \mid \mathcal{F}_{t-1}] = M_{t-1} \cdot \mathbb{E}[E_t \mid \mathcal{F}_{t-1}] \leq M_{t-1} \tag{6}$$

Thus $(M_t)$ is a non-negative supermartingale under $H_0$.

## 3 Method

### 3.1 E-values via Density Betting

We construct e-values from conformal p-values using a density-based betting framework (**??**). Before observing $p_t$, we specify a density function $\hat{f}(p)$ over $[0, 1]$ encoding our prior belief about where $p_t$ will concentrate. Any density function $\hat{f}(p)$ satisfying $\int_0^1 \hat{f}(p)\, dp = 1$ yields a valid e-value: under uniformity, $\mathbb{E}[\hat{f}(p)] = \int_0^1 \hat{f}(p)\, dp = 1$, providing a fair bet that averages to 1 under the null while allowing high payoffs when p-values concentrate.

**Proposition 2** (Density Betting Yields Valid E-values)**.** *Let $\hat{f} : [0, 1] \to [0, \infty)$ be any density function (i.e., $\int_0^1 \hat{f}(p)\, dp = 1$). If $p \sim \text{Uniform}(0, 1)$, then $e = \hat{f}(p)$ satisfies $\mathbb{E}[e] = 1$.*

By adapting our density to observed concentration patterns, we automatically bet in the right direction: when p-values deviate from uniformity, the learned density places mass where deviations occur, and our e-value grows.

PITMonitor uses a histogram density that learns from past observations:

$$\hat{f}(p) = B \cdot \frac{c_b}{\sum_j c_j} \quad \text{for } p \in \text{bin } b \tag{7}$$

where $c_b$ counts past p-values in bin $b$ and $B$ is the number of bins. The histogram is initialized with Laplace pseudocounts $c_b = 1$ for all $b$, which ensures $\hat{f}$ is a valid density from the first observation and prevents zero-count bins from generating infinite or zero e-values during early monitoring.

Under exchangeability, p-values scatter uniformly and the learned histogram spreads mass roughly evenly across bins, yielding $\mathbb{E}[e] \approx 1$. If exchangeability breaks, p-values cluster in certain bins; the histogram learns these concentration patterns and achieves $\mathbb{E}[e] > 1$, generating detection power. We update the histogram *after* computing $e_t$, ensuring $\hat{f}$ depends only on past observations, guaranteeing that it is predictable, as required for the supermartingale property of the e-process.

### 3.2 The Mixture E-process

The key challenge is that the changepoint time $\tau$ is unknown. An e-process starting at $\tau$ would be sensitive to drift beginning at $\tau$ but would miss earlier changes, while one starting too early accumulates noise that dilutes its power. Rather than commit to a single guess, we maintain a weighted mixture over all possible changepoint times:

$$M_t = \sum_{\tau=1}^{t} w_\tau \cdot M_t^{(\tau)} \tag{8}$$

where $M_t^{(\tau)} = \prod_{s=\tau}^{t} e_s$ denotes the evidence accumulated from time $\tau$ onward (defined for $\tau \leq t$), and $w_\tau = 1/(\tau(\tau+1))$ is a deterministic weight satisfying $\sum_{\tau=1}^{\infty} w_\tau = 1$. Since each component $M_t^{(\tau)}$ forms a valid e-process sensitive to drift beginning at $\tau$, the mixture is simultaneously sensitive to changepoints at any time, while remaining a valid e-process under the null by linearity of expectation.

This allows us to use an efficient recursion that avoids maintaining separate products for each $\tau$:

**Proposition 3** (Efficient Recursion)**.** *The mixture e-process satisfies:*

$$M_t = e_t \cdot (M_{t-1} + w_t) \tag{9}$$

*Proof.* Expand the definition:

$$\begin{aligned} M_t &= \sum_{\tau=1}^{t} w_\tau \cdot M_t^{(\tau)} && (10)\\ &= \sum_{\tau=1}^{t-1} w_\tau \cdot e_t \cdot M_{t-1}^{(\tau)} + w_t \cdot e_t && (11)\\ &= e_t \left( \sum_{\tau=1}^{t-1} w_\tau \cdot M_{t-1}^{(\tau)} + w_t \right) && (12)\\ &= e_t (M_{t-1} + w_t) && (13) \end{aligned}$$

□

This recursion enables an $O(1)$ update of the mixture per observation (plus $O(\log t)$ for rank computation via a sorted structure), avoiding the cost of maintaining or updating all $t$ component e-processes separately.

### 3.3 Type I Error Control

Ville's inequality (**?**) provides the anytime-valid guarantee by bounding the probability that a non-negative supermartingale *ever* exceeds a threshold, regardless of the monitoring horizon or stopping rule:

**Theorem 1** (Anytime-Valid False Alarm Control)**.** *Under $H_0$, PITMonitor satisfies:*

$$\mathbb{P}\left(\sup_{t\geq 1} M_t \geq \frac{1}{\alpha}\right) \leq \alpha \tag{14}$$

*Proof.* As shown in subsection **??**, each e-value satisfies $\mathbb{E}[e_t \mid \mathcal{F}_{t-1}] = 1$ under $H_0$.

The mixture $M_t = \sum_{\tau=1}^{t} w_\tau M_t^{(\tau)}$ is defined with $M_t^{(\tau)} = \prod_{s=\tau}^{t} e_s$ only for $\tau \leq t$. To apply Ville's inequality we work with an extended process defined for all $t \geq 0$. For each $\tau \geq 1$ define:

$$\widetilde{M}_t^{(\tau)} = \begin{cases} 1 & t < \tau \\ \prod_{s=\tau}^{t} e_s & t \geq \tau \end{cases} \tag{15}$$

Since $e_t \geq 0$ and $\mathbb{E}[e_t \mid \mathcal{F}_{t-1}] = 1$, each $(\widetilde{M}_t^{(\tau)})_{t\geq 0}$ is a non-negative martingale with $\widetilde{M}_0^{(\tau)} = 1$.

Define the full mixture over all $\tau \geq 1$:

$$\widetilde{M}_t = \sum_{\tau=1}^{\infty} w_\tau \, \widetilde{M}_t^{(\tau)} \tag{16}$$

A countable non-negative weighted combination of martingales with summable weights is itself a martingale, so $(\widetilde{M}_t)_{t\geq 0}$ is a non-negative martingale with $\widetilde{M}_0 = \sum_{\tau=1}^{\infty} w_\tau = 1$.

For $\tau > t$, $\widetilde{M}_t^{(\tau)} = 1$ since no e-values have been incorporated yet. Therefore:

$$\widetilde{M}_t = \sum_{\tau=1}^{t} w_\tau \prod_{s=\tau}^{t} e_s + \sum_{\tau=t+1}^{\infty} w_\tau \cdot 1 \tag{17}$$

$$= M_t + \sum_{\tau=t+1}^{\infty} \frac{1}{\tau(\tau+1)} \tag{18}$$

The tail sum telescopes: $\sum_{\tau=t+1}^{\infty} \frac{1}{\tau(\tau+1)} = \sum_{\tau=t+1}^{\infty} \left(\frac{1}{\tau} - \frac{1}{\tau+1}\right) = \frac{1}{t+1}$. Hence:

$$\widetilde{M}_t = M_t + \frac{1}{t+1} \geq M_t \tag{19}$$

$$\therefore \left\{\sup_{t\geq 1} M_t \geq \frac{1}{\alpha}\right\} \subseteq \left\{\sup_{t\geq 1} \widetilde{M}_t \geq \frac{1}{\alpha}\right\} \tag{20}$$

Since $(\widetilde{M}_t)$ is a non-negative martingale, Ville's inequality gives $\mathbb{P}(\sup_{t\geq 0} \widetilde{M}_t \geq 1/\alpha) \leq \alpha$, thus:

$$\mathbb{P}\left(\sup_{t\geq 1} M_t \geq \frac{1}{\alpha}\right) \leq \mathbb{P}\left(\sup_{t\geq 1} \widetilde{M}_t \geq \frac{1}{\alpha}\right) \leq \alpha \tag{21}$$

□

### 3.4 Changepoint Estimation

After an alarm at time $T$, we estimate the changepoint location by selecting the split that best explains the post-split p-values as non-uniform. For each candidate $k \in \{1, \ldots, T-1\}$, we evaluate the segment $(p_{k+1}, \ldots, p_T)$ under two hypotheses:

- $\mathbf{H_0^{(k)}}$: p-values are Uniform$(0,1)$, so each of $B$ bins receives probability $1/B$.
- $\mathbf{H_1^{(k)}}$: bin probabilities are unknown, with a symmetric Dirichlet prior Dir$(\alpha, \ldots, \alpha)$.

We use a Bayes factor rather than a likelihood ratio because the flexible model $H_1$ would otherwise trivially outperform $H_0$ by overfitting - $H_1$ contains $H_0$ as a special case, so its MLE fit is always at least as good. Averaging over the Dirichlet prior instead of optimizing penalizes $H_1$ for the prior mass it wastes on configurations the data does not support. We set $\alpha = 1/2$ (Jeffreys prior), the standard non-informative choice for categorical data, which is invariant to the reparametrization of bin boundaries (**?**).

Let $\mathbf{n} = (n_1, \ldots, n_B)$ denote the histogram of p-values in the post-split segment $(p_{k+1}, \ldots, p_T)$, where each $n_b$ counts how many p-values fell into the $b$-th equal-width bin over $[0,1)$, and $N = \sum_b n_b$ is the segment length. Under both hypotheses, the data likelihood is multinomial. Under $H_0$ every bin has probability $1/B$, so the multinomial reduces to:

$$\log p(\mathbf{n} \mid H_0) = \log \frac{N!}{\prod_b n_b!} - N \log B \tag{22}$$

Under $H_1$, the bin probabilities $\theta$ are unknown so we integrate them out over the Dirichlet prior:

$$\begin{aligned} \log p(\mathbf{n} \mid H_1) = \log \frac{N!}{\prod_b n_b!} &+ \log \Gamma(B\alpha) - \log \Gamma(N + B\alpha) \\ &+ \sum_{b=1}^{B} \big[\log \Gamma(n_b + \alpha) - \log \Gamma(\alpha)\big] \end{aligned} \tag{23}$$

Since the combinatorial factor appears in both likelihoods, it cancels in the log Bayes factor:

$$\begin{aligned} \log \mathrm{BF}_k &= \log p(\mathbf{n} \mid H_1) - \log p(\mathbf{n} \mid H_0) \\ &= \log \Gamma(B\alpha) - \log \Gamma(N + B\alpha) \\ &\quad + \sum_{b=1}^{B} \big[\log \Gamma(n_b + \alpha) - \log \Gamma(\alpha)\big] \\ &\quad + N \log B \end{aligned} \tag{24}$$

To identify our changepoint, we simply select $\hat{\tau} = \arg\max_k \log \mathrm{BF}_k$.

This changepoint estimate is a capability absent from all `river` baselines, which expose only a binary alarm flag. PITMonitor thus enables practitioners not just to detect that drift has occurred, but to identify how far back model outputs were already corrupted - directly actionable for deciding how much historical inference to distrust or recompute.

**Algorithm 1** PITMonitor

**Require:** Significance level $\alpha$, number of bins $B$

Initialize: $M_0 \leftarrow 0$, histogram counts $c_1, \ldots, c_B \leftarrow 1$ ▷ Laplace prior
**for** $t = 1, 2, \ldots$ **do**
  Observe PIT $U_t \in [0, 1]$
  Insert $U_t$ into sorted list; compute rank $R_t$
  Sample $V_t \sim \text{Uniform}(0, 1)$
  $p_t \leftarrow (R_t - 1 + V_t)/t$ ▷ Conformal p-value
  $b \leftarrow \lfloor p_t \cdot B \rfloor + 1$ ▷ Histogram bin index
  $e_t \leftarrow B \cdot c_b / \sum_{j=1}^{B} c_j$ ▷ E-value from density
  $c_b \leftarrow c_b + 1$ ▷ Update histogram after computing $e_t$
  $w_t \leftarrow 1/(t \cdot (t + 1))$ ▷ Deterministic mixture weight
  $M_t \leftarrow e_t \cdot (M_{t-1} + w_t)$ ▷ Mixture e-process
  **if** $M_t \geq 1/\alpha$ **then**
    **return** ALARM at time $t$
  **end if**
**end for**

### 3.5 Complete Algorithm

## 4 Experiments

We evaluate PITMonitor on the `river` FriedmanDrift benchmark, a standard regression task for evaluating concept drift detectors under controlled conditions, comparing against the seven included stream-drift detectors from the `river` library.

### 4.1 Setup

**Dataset and drift scenarios.** FriedmanDrift (**?**) is a synthetic regression stream with 10 input features ($x_0$–$x_9$). Only features $x_0$–$x_4$ appear in the true function; $x_5$–$x_9$ are noise. We evaluate three drift types that represent qualitatively different distribution changes:

- **GRA** (Global Recurring Abrupt, $\Delta t = 0$): All relevant features change simultaneously at an abrupt onset. This is the canonical detection scenario.
- **GSG** (Global Slow Gradual, $\Delta t = 500$): The change spreads linearly across all features over a 500-sample transition window, representing gradual covariate shift.
- **LEA** (Local Expanding Abrupt): Drift starts on a subset of features and expands to include more over time, representing localized distribution change.

**Stream layout.** Each stream consists of three contiguous segments: $n_{\text{train}} = 10{,}000$ pre-drift samples for model training, $n_{\text{stable}} = 2{,}500$ pre-drift monitoring samples that define the null-hypothesis window for FPR estimation, and $n_{\text{post}} = 2{,}500$ post-drift samples for TPR estimation. The drift onset occurs at the boundary between the stable and post-drift segments.

**Predictive model.** We train a **ProbabilisticMLP**: a feedforward neural network outputting a Gaussian predictive distribution $\mathcal{N}(\mu_t, \sigma_t^2)$ for each input. The network has 3 hidden layers of 128 units with SiLU activations. Inputs and targets are standardized using per-feature means and standard deviations fitted on the training set. Training uses mini-batches of size 256, the Adam optimizer with initial learning rate $3 \times 10^{-4}$, a cosine annealing learning rate schedule, and 500 epochs. The model achieves $R^2 = 0.96$ on a held-out pre-drift test set, with an expected calibration error (ECE) of 0.01, confirming it is well-specified and calibrated before monitoring begins. We define ECE as the average absolute deviation between the empirical PIT CDF and the identity line over a uniform grid on $[0, 1]$: $\text{ECE} = \frac{1}{G}\sum_{g=1}^{G}\left|\hat{F}_{\text{PIT}}(p_g) - p_g\right|$, where $p_g = g/G$ for $G = 100$ grid points. Under perfect calibration (PITs exactly uniform), ECE $= 0$.

**PIT construction.** For each monitoring sample $(x_t, y_t)$ the PIT is:

$$U_t = \Phi\left(\frac{y_t - \mu_t}{\sigma_t}\right) \tag{25}$$

where $\mu_t$, $\sigma_t$ are the predicted mean and standard deviation and $\Phi$ is the standard normal CDF.

**Detector settings.** All `river` baselines are run with their library-default parameters. This is the most defensible comparison: it reflects the out-of-the-box experience a practitioner receives, avoids any implicit tuning in favour of a particular detector, and sidesteps the need for held-out null data or advance knowledge of the monitoring window length. Aligning a baseline's internal parameters with a target stream-level FPR would require one of these — exactly what PITMonitor obviates.

- **PITMonitor**: $\alpha = 0.05$, $B = 100$. A key practical advantage is that PITMonitor has only two interpretable parameters: $\alpha$ directly and provably controls the stream-level false alarm probability (Theorem **??**), and $B$ controls the histogram resolution. No pre-specified monitoring horizon, held-out null data, or calibration step is required.
- **Continuous-input baselines (ADWIN, KSWIN, PageHinkley)**: library-default parameters. Their internal sensitivity parameters (ADWIN's $\delta$, KSWIN's significance level, PageHinkley's threshold) have indirect, horizon-dependent relationships to stream-level FPR.
- **Binary-input baselines (DDM, EDDM, HDDM_A, HDDM_W)**: library-default parameters. These detectors require a binary error signal; we binarize via $b_t = \mathbf{1}[|r_t| > \theta]$ where $\theta$ is the *median* of $|r_t|$ on the training data (giving $\approx 50\%$ pre-drift error rate). The median is the most neutral, assumption-free threshold choice: it makes no assumption about the detectors' preferred operating error rate. That binary detectors were designed for lower error rates ($\sim$ 5–20%) is an inherent limitation of applying them to regression monitoring, and one that PITMonitor avoids by consuming PIT values directly.

**Baselines.** We compare against seven stream-drift detectors from `river`. Different detector families consume different input streams, reflecting their design assumptions:

- **Continuous input** (squared residuals $r_t^2 = (y_t - \mu_t)^2$): ADWIN, KSWIN, PageHinkley. These methods track a running statistic of the input stream and alarm when it changes significantly.

Table 1: Drift detection results on FriedmanDrift (1,000 trials, $\alpha = 0.05$). Mean delay is in samples. Best TPR per scenario is **bolded**; FPR exceeding $\alpha$ is underlined.

| | GRA | | | GSG | | | LEA | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | TPR | FPR | Delay | TPR | FPR | Delay | TPR | FPR | Delay |
| PITMonitor | 95.7% | 4.3% | 76 | 95.7% | 4.3% | 189 | 0.0% | 4.3% | — |
| ADWIN | **99.0%** | 1.0% | 27 | **99.0%** | 1.0% | 27 | **99.0%** | 1.0% | 118 |
| KSWIN | 2.9% | 97.1% | 17 | 2.3% | 97.7% | 47 | 3.3% | 96.7% | 571 |
| PageHinkley | 0.3% | 99.7% | 1 | 0.3% | 99.7% | 11 | 0.3% | 99.7% | 49 |
| DDM | 92.1% | 7.9% | 407 | 90.8% | 7.9% | 658 | 2.0% | 7.9% | 1129 |
| EDDM | 9.7% | 90.3% | 349 | 9.7% | 90.3% | 573 | 0.4% | 90.3% | 1166 |
| HDDM_A | 94.7% | 5.3% | 60 | 94.7% | 5.3% | 170 | 6.1% | 5.3% | 1183 |
| HDDM_W | 10.7% | 89.3% | 14 | 10.7% | 89.3% | 48 | 10.6% | 89.3% | 584 |

- **Binary input** (error indicator $b_t = \mathbf{1}[|r_t| > \theta]$ where $\theta$ is the median of $|r_t|$ on the training data, giving $\approx 50\%$ pre-drift error rate): DDM, EDDM, HDDM_A, HDDM_W. These methods are designed for binary error streams and alarm when the error rate increases.

PITMonitor receives PIT values; all baselines receive the appropriate residual-derived signal. This is the fairest comparison: each method gets the input it was designed for.

**Evaluation protocol.** We run $N = 1{,}000$ Monte Carlo trials per scenario, each using a distinct random seed for the data stream. For each trial we record whether an alarm fires, its index, and whether it occurred before or after the true drift onset, as well as distance from the true changepoint for PITMonitor only. We report:

- **TPR**: fraction of trials with a true-positive alarm (alarm fired after drift onset).
- **FPR**: fraction of trials with a false alarm (alarm fired before drift onset).
- **Mean detection delay**: mean number of samples between the true drift onset and the alarm, over true-positive trials only.

### 4.2 Results

Table **??** presents the full results. Figure **??** visualizes TPR and FPR per method and scenario, Figure **??** shows a representative single-run monitoring trace for the GRA scenario, and Figure **??** shows the distribution of detection delays.

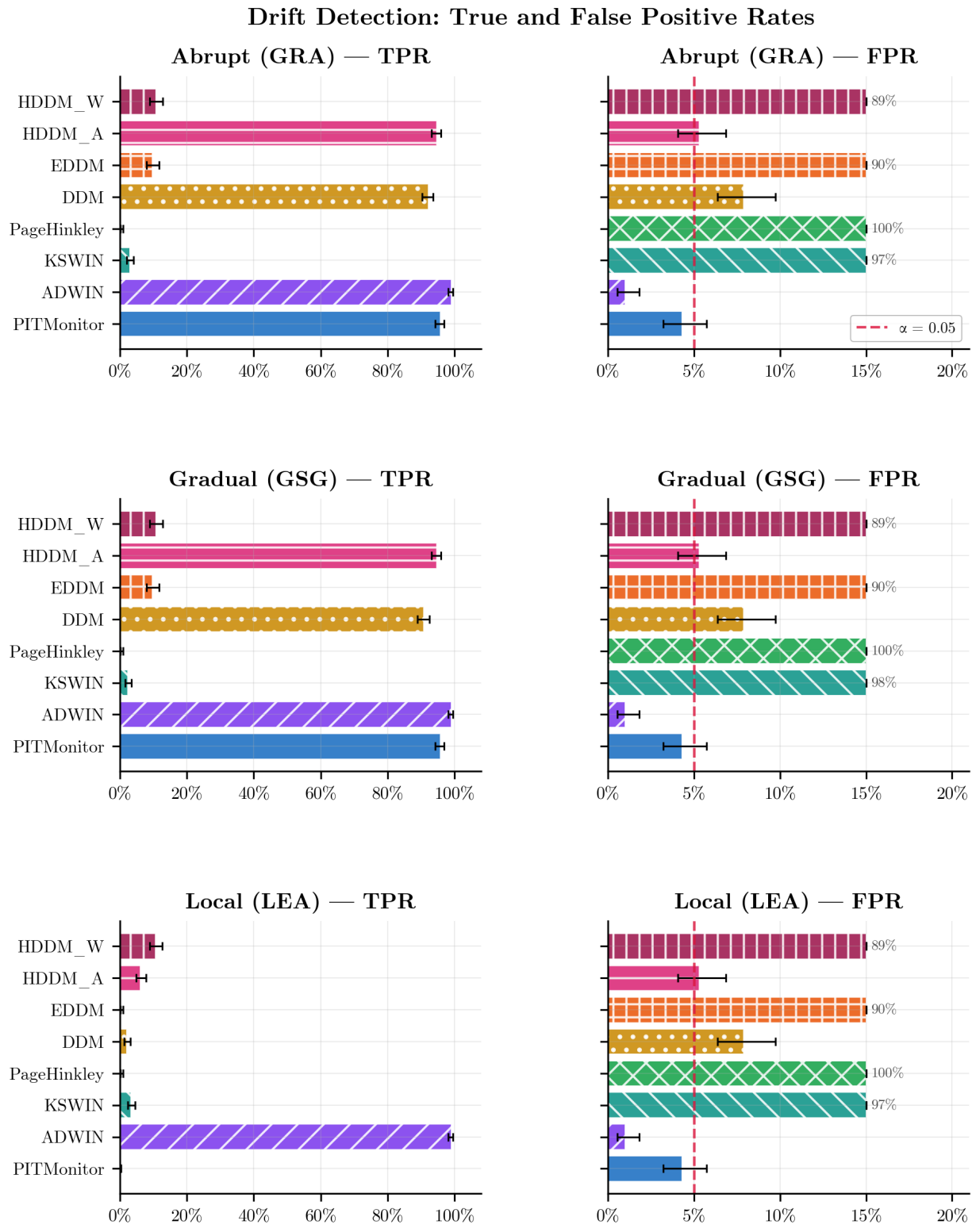


Figure 1: TPR and FPR across all detectors and drift scenarios. The red dashed line marks $\alpha = 0.05$. River baselines are run at library defaults; PITMonitor is the only method with a proven stream-level FPR guarantee (Theorem **??**).

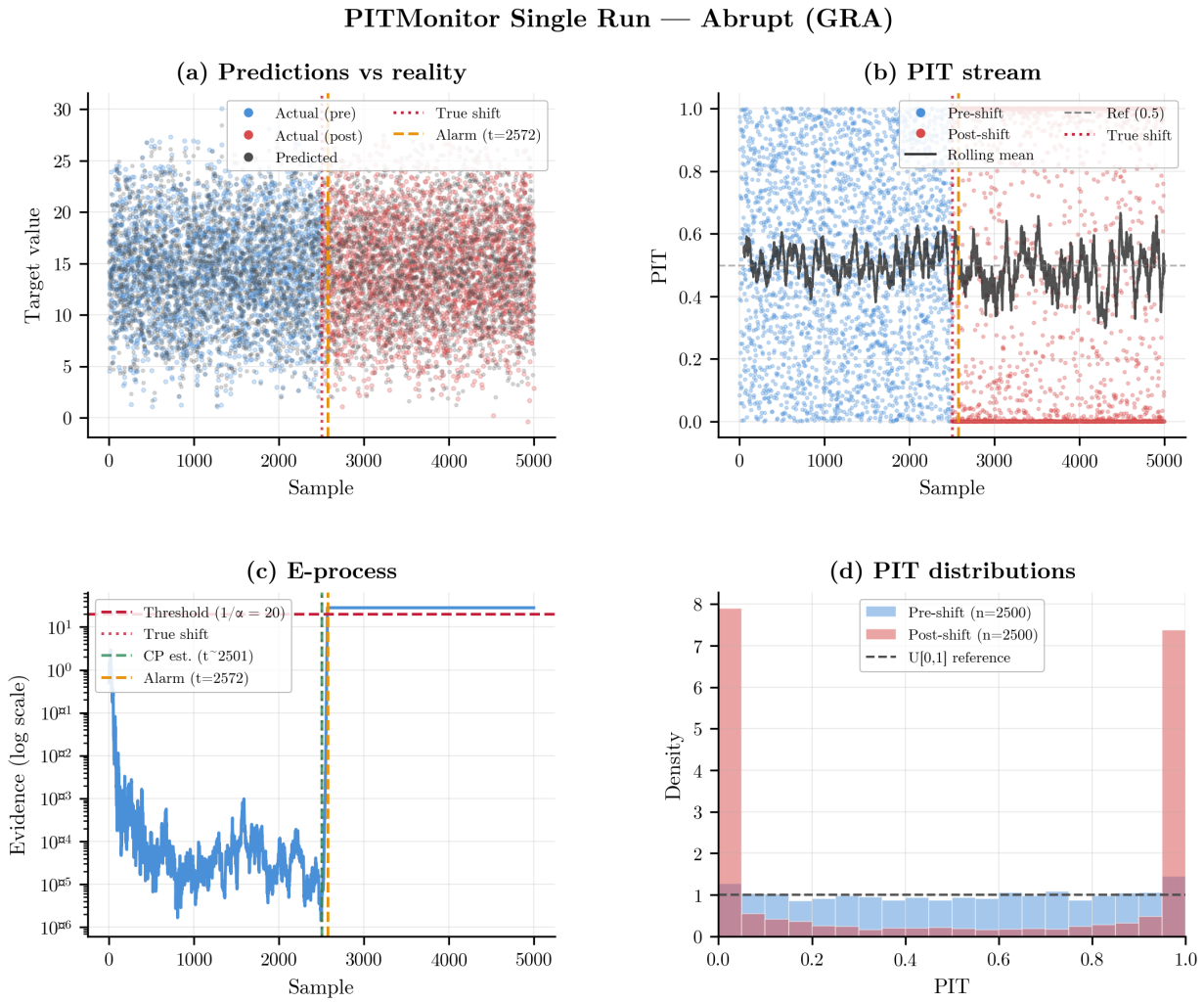


Figure 2: Single-run PITMonitor trace on the GRA (abrupt) scenario. (a) Predicted vs. actual values with alarm marker. (b) PIT stream with 50-sample rolling mean. (c) Mixture e-process on a log scale; the dashed red line is the alarm threshold $1/\alpha = 20$. (d) Pre- and post-shift PIT histograms illustrating the calibration change detected by the algorithm.

**Type I error control.** PITMonitor achieves FPR of 4.3% across all scenarios, consistent with the nominal $\alpha = 0.05$ guarantee proved in Theorem **??**. This is the only detector in the comparison for which FPR control is formally guaranteed regardless of monitoring horizon. ADWIN maintains 1.0% empirical FPR on the 2,500-sample null window; however, this figure is specific to that window length. Because ADWIN evaluates its Hoeffding bound at every observation, false alarm opportunities accumulate with the monitoring horizon: a practitioner monitoring over 25,000 samples at the same $\delta$ setting could expect substantially more spurious alarms than observed here. PITMonitor's $\mathbb{P}(\text{ever alarm} \mid H_0) \leq \alpha$ bound holds regardless of how long monitoring continues. Among the remaining baselines, HDDM_A maintains moderate FPR, while KSWIN, PageHinkley, EDDM, and HDDM_W exhibit FPR exceeding 89% — these detectors alarm on nearly every null run, rendering them unusable for practical monitoring without re-tuning. This illustrates the fundamental challenge of deploying library-default baselines: parameters tuned for one setting (e.g.,

very short streams or very large drifts) can fail catastrophically in others.

**Global drift (GRA, GSG).** PITMonitor achieves 95.7% TPR with 4.3% FPR on both global drift scenarios, with detection delay of 76 samples on GRA and 189 on GSG. ADWIN achieves 99.0% TPR with 1.0% FPR and a mean delay of 27 samples on both GRA and GSG; the identical delay across these scenarios reflects that ADWIN's Hoeffding bound is sensitive enough to detect even the early portion of the gradual transition window. Among the binary baselines, DDM and HDDM_A achieve 92.1% and 94.7% TPR respectively; DDM's mean delay of 407 samples on GRA reflects slower adaptation on the binarized error stream. The remaining detectors (KSWIN, PageHinkley, EDDM, HDDM_W) have near-zero TPR as a direct consequence of their high FPR: having already fired during the null window, they are never credited with a true-positive detection.

**Local expanding drift (LEA).** PITMonitor achieves 0.0% TPR on the LEA scenario, correctly maintaining FPR control but failing to detect the drift. This reveals a meaningful limitation: LEA drift begins on a subset of the five relevant features, producing a small initial perturbation to the Gaussian predictive distribution that the histogram e-value does not accumulate evidence for quickly enough within the 2,500-sample post-drift window. ADWIN's squared-residual statistic is more sensitive to partial distributional shifts than PITMonitor's PIT-based approach.

**Detection delays.** Figure **??** shows the full distribution of detection delays across true-positive trials. PITMonitor's delay distribution is unimodal and moderately concentrated. Its mean delay of 76 samples on GRA reflects the amount of evidence the mixture e-process must accumulate before crossing $1/\alpha = 20$; this is the explicit cost of the anytime-valid guarantee — stricter evidentiary requirements mean the process is more conservative by design, not a failure of sensitivity. ADWIN's distribution reflects the Hoeffding bound of its adaptive window; its mean delay is identical on GRA and GSG because its sensitivity at library defaults is sufficient to detect even the earliest portion of the gradual transition window. HDDM_A achieves shorter delay than PITMonitor on GRA (60 vs 76 samples) but lacks any formal FPR guarantee.

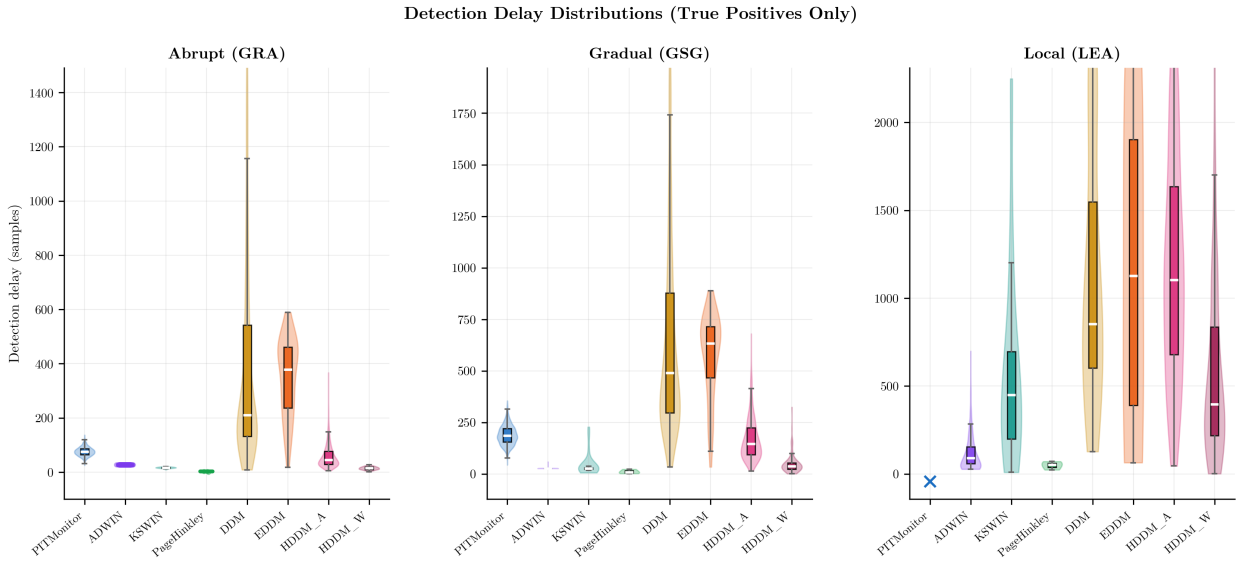


Figure 3: Detection delay distributions (true-positive trials only) across all detectors and drift scenarios. Methods with zero true-positive detections are marked with × at the bottom. Violins show the full delay distribution; the white bar marks the median.

**Changepoint localization.** A unique property of PITMonitor is the Bayes-factor changepoint estimate available after each alarm. None of the `river` baselines provide an analogous estimate; they expose only a binary alarm flag. Figure **??** shows the distribution of absolute changepoint error $|\hat{\tau} - \tau|$ across true-positive trials. PITMonitor's mean absolute changepoint error is 1.1 samples on GRA and 6.9 on GSG, demonstrating that the estimated changepoint closely tracks the true onset. This provides the practitioner with a starting point for diagnosing and correcting the drift.

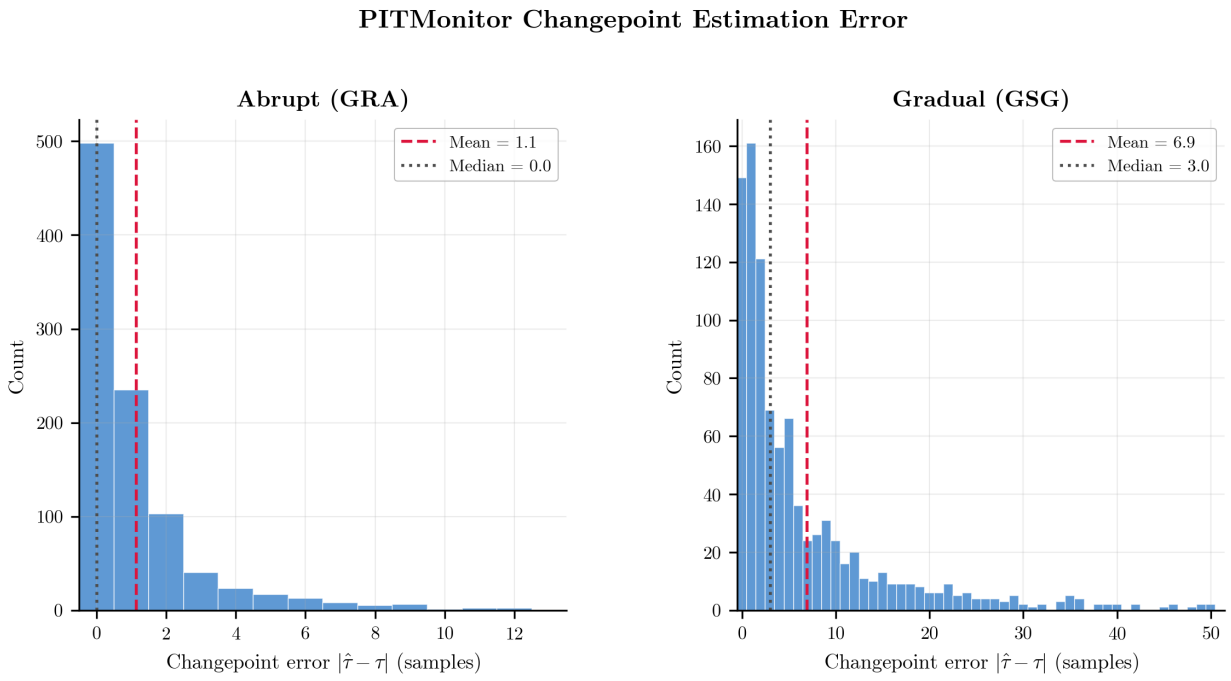


Figure 4: Distribution of PITMonitor changepoint estimation error $|\hat{\tau} - \tau|$ across true-positive trials. The abrupt (GRA) scenario yields highly concentrated estimates; the gradual (GSG) scenario shows slightly larger errors due to the extended transition window.

## 5 Discussion

**When to use PITMonitor.** PITMonitor is designed for continuous monitoring of deployed probabilistic models where:

- False alarms have real costs (unnecessary retraining, alert fatigue, loss of trust)
- The monitoring horizon is indefinite or stopping is data-dependent
- *Calibration* drift — changes in predictive uncertainty — is the primary concern, rather than raw accuracy degradation; residual-based detectors may miss a model that becomes systematically overconfident while maintaining prediction accuracy
- *Global* calibration drift, not localized feature degradation, is the concern
- Accurate changepoint localisation is valuable

For one-time calibration assessment ("is this model calibrated?"), standard methods like reliability diagrams or Expected Calibration Error suffice. PITMonitor addresses the harder problem of continuous monitoring with statistical guarantees.

**Limitations.** *Local and partial drift.* The LEA results show PITMonitor has low power when drift is initially confined to a subset of features. When only some input directions shift, the predictive

Gaussian may change in mean or variance only slightly, producing weak signal in the PIT sequence. ADWIN on squared residuals can more directly detect a mean shift in prediction error regardless of how many features are responsible. Practitioners monitoring models where feature-level drift is anticipated may need to complement PITMonitor with feature-level monitors.

*Detection delay vs. FPR control.* PITMonitor's anytime-valid guarantee comes at the cost of later detection compared to ADWIN. The mean delay on GRA (76 samples) reflects the number of post-drift observations needed for the e-process to cross $1/\alpha = 20$. The most direct way to reduce delay is to accept a larger $\alpha$: setting $\alpha = 0.10$ lowers the threshold to 10, requiring less accumulated evidence before an alarm fires. Users should calibrate $\alpha$ against their tolerance for false alarms over the full expected deployment horizon rather than treating it as a fixed convention.

*Exchangeability assumption.* PITMonitor tests exchangeability of PITs. If pre-change PITs exhibit temporal dependence (e.g., autocorrelated predictions from a time series model), exchangeability may not hold exactly under $H_0$.

*n_bins sensitivity.* The number of histogram bins $B$ controls the bias-variance tradeoff in the density estimator. We use $B = 100$ for these experiments; empirically this provides good adaptation speed with stable FPR. Smaller $B$ is more stable but slower to adapt; larger $B$ adapts faster but at the cost of more variance in the estimated density.

**Practical recommendations.**

- Set $\alpha$ based on tolerance for false alarms over the deployment horizon. For safety-critical systems, $\alpha = 0.01$ may be appropriate; for exploratory monitoring, $\alpha = 0.10$ allows faster detection.
- Use $B = 100$ histogram bins as a default for large monitoring windows; reduce to $B = 10$–$20$ for smaller windows ($n_{\text{monitor}} < 500$).
- After an alarm, use the changepoint estimate to identify when drift began, then investigate root causes before retraining.
- For models where localized feature drift is anticipated, consider running PITMonitor in parallel with ADWIN on squared residuals. PITMonitor provides the changepoint localisation and anytime-valid FPR control; ADWIN provides faster detection of partial shifts, accepting a higher false alarm rate.

**Comparison to `river` baselines.** The experiments highlight two structural dimensions on which PITMonitor differs from the `river` baselines.

*Parameter simplicity and FPR guarantees.* PITMonitor has two interpretable parameters: $\alpha$, which directly and provably controls stream-level FPR for any monitoring horizon, and $B$, the histogram resolution. The `river` baselines expose multiple internal parameters whose relationship to stream-level FPR is indirect, horizon-dependent, and not formally guaranteed. Even ADWIN's $\delta$ — the most interpretable among them — bounds per-comparison error, not per-stream error, so achieving a target FPR requires advance knowledge of the monitoring window length or empirical calibration. Binary baselines additionally require a binarization threshold with no guidance from the library.

*Detection power.* On global drift (GRA, GSG), ADWIN achieves 99.0% TPR with mean delay 27 samples at FPR 1.0%; HDDM_A achieves 94.7% TPR with delay 60 samples at FPR 5.3%. PIT-

Monitor achieves 95.7% TPR with 76-sample delay at 4.3% FPR, with the unique property that its FPR bound holds anytime. On local expanding drift (LEA) ADWIN is dominant while PITMonitor has essentially zero power. KSWIN, PageHinkley, HDDM_W, and EDDM are effectively inoperable at library defaults in this setting: their FPR exceeds 89%, meaning they alarm during the null window on nearly every trial and thus cannot register a true-positive detection. Their near-zero TPR is a consequence of their high FPR, not of their detection mechanism per se.

*Calibration specificity.* PITMonitor detects changes in the PIT distribution, which encodes the full calibration behaviour of the predictive model: systematic over- or under-confidence, shifts in predictive mean, and changes in predictive variance all manifest as non-uniformity or non-stationarity in the PIT stream. The `river` baselines operate on squared residuals — an accuracy proxy — which means calibration changes that leave point-prediction accuracy unchanged may go undetected. A model whose predictions become systematically overconfident (confidence intervals that are too narrow) without shifting its conditional mean would produce unchanged squared residuals but strongly non-uniform PITs. PITMonitor is the only method in the comparison designed to detect exactly this class of calibration failure.

*Benchmark context.* ADWIN's strong performance on FriedmanDrift should be interpreted in context: FriedmanDrift is `river`'s own canonical synthetic benchmark, and ADWIN's library defaults were likely developed and validated on comparable data. Strong empirical results on a method's home benchmark do not straightforwardly generalise to the diversity of real-world monitoring scenarios. Moreover, ADWIN's 1.0% empirical FPR on the 2,500-sample null window does not constitute a formal guarantee: deployments with longer monitoring horizons will accumulate additional Hoeffding comparisons, increasing the stream-level false alarm probability beyond what this experiment demonstrates.

# 6 Related Work

### Calibration Assessment

Classical calibration metrics include Expected Calibration Error (**?**), reliability diagrams (**?**), and proper scoring rules (**?**). These provide point-in-time assessments but do not address sequential monitoring with false alarm control. PITs have been used for forecast evaluation in econometrics (**?**) and weather prediction (**?**).

### Distribution Shift Detection

Methods for detecting covariate shift include two-sample tests (**?**), domain classifiers (**?**), and conformal approaches (**?**). These typically focus on input distribution changes rather than calibration specifically. Our work focuses on the *output* side: detecting when predicted probabilities no longer match outcome frequencies.

### Sequential Calibration Testing

**?** proposed e-values for testing forecast calibration, focusing on whether PITs are uniform. Our work differs in two ways: (1) we test exchangeability rather than uniformity, enabling insensitivity

to i.i.d. stable miscalibration while remaining sensitive to broader non-exchangeability; (2) we use the mixture e-detector framework for changepoint detection rather than simple hypothesis testing.

### E-values and Anytime-Valid Inference

The e-value framework has seen rapid development (**???**). Applications include A/B testing (**?**), clinical trials (**?**), and conformal prediction (**?**). The e-detector framework for changepoint detection was introduced by **?**, providing the theoretical foundation for our mixture e-process.

### Changepoint Detection

Classical methods include CUSUM (**?**) and Shiryaev-Roberts procedures (**??**). These typically assume known pre- and post-change distributions. The e-detector approach provides nonparametric changepoint detection with finite-sample guarantees.

## 7 Conclusion

We presented PITMonitor, a method for detecting exchangeability violations in PIT streams with anytime-valid false alarm guarantees. By testing exchangeability of probability integral transforms using a mixture e-process, PITMonitor enables continuous monitoring without inflating Type I error, regardless of when or why monitoring stops.

Experiments on three FriedmanDrift scenarios, with all baselines run at library defaults, demonstrate that PITMonitor achieves competitive detection performance on global drift (GRA and GSG) — 95.7% TPR with 4.3% FPR and a unique changepoint localization capability absent from all `river` baselines — while providing the only formally proven stream-level FPR guarantee (Theorem **??**). The results also surface an honest limitation: PITMonitor has zero detection power on local expanding drift (LEA), while ADWIN detects this scenario with 99.0% TPR at 1.0% FPR.

Two structural advantages persist beyond this comparison. First, PITMonitor's FPR guarantee is *horizon-independent*: whereas ADWIN's empirical false alarm rate grows with the number of observations monitored (each new sample triggering an additional Hoeffding comparison), PITMonitor provably satisfies $\mathbb{P}(\text{ever alarm} \mid H_0) \leq \alpha$ regardless of deployment duration. Second, PITMonitor targets *calibration specifically*: by operating on probability integral transforms rather than squared residuals, it is sensitive to changes in predictive uncertainty — systematic overconfidence, underconfidence, or variance shifts — that leave accuracy metrics unchanged and would be missed entirely by residual-based detectors.

PITMonitor requires only two interpretable parameters ($\alpha$ and $B$) with no pre-specified monitoring horizon, held-out null data, or binarization step required. Practitioners should view it as the right tool for global calibration monitoring with strict FPR control, and consider complementing it with simpler residual-based detectors when feature-level drift is anticipated.

Future work includes extensions to temporally dependent predictions, multivariate outputs, and improving power on partial distributional shifts.

**Code Availability.** PITMonitor is available at `https://github.com/tristan-farran/pitmon`.