Entropy-Driven Change-Point Detection via Bayesian Methods

Bohan Zhu

1 Introduction

Time series analysis is a central area in statistics, with broad applications across fields such as economics, public health, and environmental science. One key focus within this area is change point detection, which seeks to identify moments when the statistical properties of a sequence shift in a meaningful way. In real-world scenarios, detecting these shifts can serve as an early warning signal for upcoming transitions. For example, sudden changes in financial indicators prior to the 2008 global financial crisis offered signals of market instability. Similarly, in public health, abrupt changes in disease incidence may signal the emergence of an outbreak[1].

Detecting such change points is critical for understanding dynamic behavior, identifying phase transitions, and building models that reflect different system phases[2].

In this study, we construct a controlled simulation framework based on a two-dimensional trajectory. The system begins with stable fluctuations and then undergoes a structural change at an unknown point in time. The trajectory takes the form of a moving point in space, designed to mimic the evolution of a system state over time.

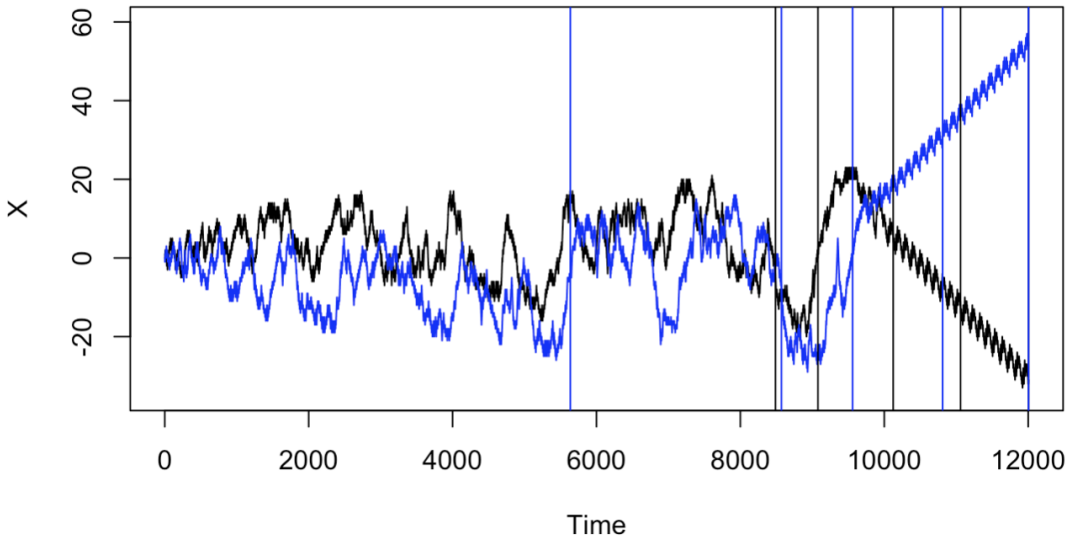
Our goal is to identify true change points in such sequences—distinguishing meaningful structural changes from short-term noise or random variation. To achieve this, we apply and compare several statistical approaches, including both traditional and Bayesian methods, and evaluate their performance under this well-defined setting.

2 Classical Methods for Change-Point Detection

To establish a baseline, we first apply classical changepoint detection methods to the original ant trajectory data, implemented using the `changepoint` package in R [3]. These methods identify structural changes in the data based on abrupt shifts in statistical properties such as the mean or variance.

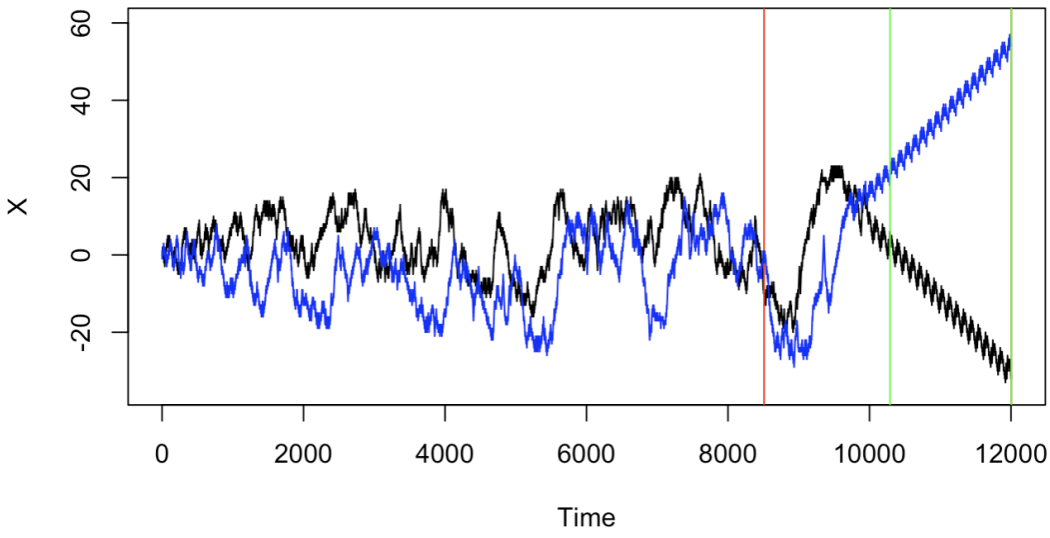
We begin by detecting changes in the mean using the `cpt.mean()` function from the `changepoint` package. This method segments the time series by locating points where the mean value significantly shifts. In our implementation, we use the Segment Neighborhood (SegNeigh) method with the AIC (Akaike Information Criterion)[4]penalty to balance model complexity and fit. The detected changepoints are marked with vertical lines on the X and Y lines.

Figure 1: Change-Point Detection Based on Mean Shifts



Next, we explore changes in variance using `cpt.var()`, applying the AMOC (At Most One Change) method with the same AIC penalty. This allows us to capture abrupt changes in data spread, which may signal behavioral transitions not reflected in mean shifts. Again, we visualize the detected changepoints for both spatial dimensions, highlighting the structural shifts with vertical markers.

Figure 2: Change-Point Detection Based on Variance Shifts



3 Bayesian Methods as a Remedy for Key Limitations

Traditional change-point detection methods, such as those based on shifts in means or variances, are simple to implement and computationally efficient. However, they face several key limitations that can hinder their robustness and interpretability.

First, many classical methods tend to identify only a single most likely change-point without addressing the uncertainty around that estimate. This non-uniqueness issue means that repeated runs or small changes in tuning parameters can yield different change-point results. While this flexibility may be useful for exploration, it also leads to ambiguity in interpretation. In contrast, Bayesian methods not only give a full distribution over possible change-points, but also often concentrate the probability on a single most plausible location.

Second, these approaches often rely on hand-tuned thresholds or heuristic rules. These criteria, although intuitive, lack a principled statistical foundation for evaluating how confident we should be in the detected change-points. This can raise concerns about the reproducibility and statistical significance of the results.

Bayesian methods provide a structured alternative. Rather than producing a single deterministic estimate, they model unknown quantities as probability distributions and update these beliefs as new data arrives. This allows us to assign probabilities to different change-point locations and quantify our uncertainty about them. Moreover, by using priors and methods like MCMC sampling, Bayesian approaches can integrate over many models and reduce sensitivity to arbitrary tuning decisions[5].

Finally, because Bayesian methods generate posterior distributions instead of relying on fixed thresholds, we can directly assess how confident we are in each detected change-point, improving both interpretability and statistical reliability.

4 The BOCPD Framework

In Bayesian inference, any probabilistic update generally follows three key components:

- Likelihood: How well the observed data fits the model under a given parameter.
- Prior: Our belief before seeing the data.
- Posterior: The updated belief after observing the data.

Bayesian Online Change Point Detection (BOCPD) follows this same structure, but applies it sequentially over time[6].

We implement the BOCPD method using a custom function called `bocpd()`, written in Rcpp, a package that allows efficient C++ code to run inside R[7]. The `bocpd()` function takes the observed data and returns a posterior probability vector `prob`, where each `prob`$[i]$ represents the probability that a change point occurs at time step i.

At each step, `prob`$[i]$ is updated recursively through three components:

1. The likelihood based on the current data point;
2. A prior update that incorporates the hazard rate;
3. A posterior update that combines the two:

$$\text{prob}_i \propto \text{likelihood}_i \times \text{prior}_i$$

In our Bayesian change-point detection framework, we do not apply the models directly to the raw categorical state sequence. Instead, we first transform the sequence into a numerical indicator that reflects local uncertainty—Shannon entropy, defined as

$$H = -\sum p_i \log p_i,$$

where p_i is the empirical probability of each observed state in a sliding window. This entropy measure captures the degree of disorder or unpredictability in a categorical system and serves as an early warning signal for regime shifts[8].

Shannon entropy is chosen for its simplicity, interpretability, and wide usage in time series analysis[9] . Unlike alternative formulations (e.g., Renyi or Tsallis entropy), it does not require tuning additional parameters and assumes minimal prior structure, making it ideal for exploratory modeling.

To further stabilize the entropy sequence and reduce short-term fluctuations caused by windowing, we apply a moving average smoothing step before change-point detection[10]. This method averages adjacent values over a fixed window, helping to highlight underlying structural changes while suppressing local noise. Its simplicity and effectiveness make it suitable for preprocessing in change-point analysis.

The smoothed entropy sequence then becomes the modeling target in both our BOCPD and MCMC-based Bayesian procedures described below.

4.1 Likelihood

The likelihood measures how likely the current value x_i is, assuming no change has happened yet. We assume the data follows a normal distribution with fixed variance, so the likelihood at time i is:

$$\text{likelihood}_i = \exp\left(-\frac{(x_i - \mu_{i-1})^2}{2\sigma^2}\right)$$

where μ_{i-1} is the running mean up to time $i-1$ and σ^2 is the fixed variance

This captures how well x_i fits the expected pattern based on past data. We assume a normal distribution based on the central limit theorem. Since entropy values are computed as moving summaries over time windows, they tend to approximate normality when the window size is large enough. This makes the Normal model a reasonable and convenient choice for estimating likelihood.

This gives us the term $P(x_t \mid r_t, x_{1:t-1})$, which tells us how likely we are to observe the new data point x_t, given the current run length and all past observations.

4.2 Background on Hazard Function and Survival Analysis

In survival analysis, the hazard function $h(t)$ describes the instantaneous probability that an event will occur at time t, given that it has not occurred before. It is defined as:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t}$$

This represents the chance of "dying" in the next instant, assuming survival up to time t. In our case, each data segment is treated as something that "survives" until a change point ends it. So detecting a change point is like detecting the "death" of a segment.

To simplify the model, we assume the hazard function is constant:

$$h(t) = \lambda, \quad \text{for all } t$$

This means the probability of a change point occurring at any time is constant and equal to λ.

4.3 Prior

In BOCPD, the prior reflects our belief about whether a change point will occur at the next time step. Based on the constant hazard assumption, the probability of a change point at time t is:

$$P(\text{change at } t) = \lambda$$

And the probability of no change is:

$$P(\text{no change at } t) = 1 - \lambda$$

This leads to the prior update rule in the BOCPD recurrence:

$$\text{prior}_i = (1 - \lambda) \cdot \text{prob}[i - 1] + \lambda$$

This corresponds to the transition probability $P(r_t \mid r_{t-1})$, which depends only on how long the current segment has lasted. This update combines two possibilities: continuing the same segment (with probability $1 - \lambda$) or starting a new one (with probability λ).

4.4 Posterior

The posterior is the final output of each time step. It reflects how likely it is that a change point occurs at time t, combining both how well the current observation fits the expected pattern (likelihood) and how likely a change was before seeing this data point (prior).

More formally, the Bayesian update at time t is written as:

$$P(r_t \mid x_{1:t}) \propto P(x_t \mid r_t, x_{1:t-1}) \cdot P(r_t \mid r_{t-1})$$

In this expression, r_t is the current run length, which counts how many steps it's been since the last change point. The sequence $x_{1:t}$ includes all observed data up to time t. The first term on the right is the likelihood, and the second term is the prior based on the previous run length.

We do not normalize the posterior at each step, since the relative comparisons are sufficient to detect change points. A full normalization can be applied at the final step if needed. This way, we can still compare posterior values across time and look for peaks that indicate likely change points.

5 Bayesian Inference on Entropy Signals

Building on the smoothed Shannon entropy sequence introduced earlier, we now apply the BOCPD model to generate posterior probabilities of change-points across time. This section describes the implementation steps and outputs of the procedure.

5.1 Input Sequence

We apply the BOCPD framework to a one-dimensional time series derived from the system state. In our case, we use the smoothed entropy sequence, denoted as $entropy_X$. This sequence was originally computed as part of the machine learning method. It summarizes local uncertainty by calculating windowed entropy over recent categorical states. While it was first used in a threshold-based procedure, here we reuse it as the input to a Bayesian change point detection procedure.

5.2 Run the BOCPD algorithm

We apply the custom `bocpd()` function to the entropy sequence. This function implements the recursive update described in Section 3.1 and returns a posterior probability at each time point.

We pass the entropy series as input, along with a fixed hazard rate λ. The output is a vector `prob`, where each entry represents the (unnormalized) posterior probability that a change point occurs at that position.

Formally, for each time i, the posterior is computed as:

$$\text{prob}_i \propto \exp\left(-\frac{(x_i - \mu_{i-1})^2}{2\sigma^2}\right) \cdot \left[(1-\lambda) \cdot \text{prob}_{i-1} + \lambda\right]$$

The posterior computation step transforms the original time series into a sequence of Bayesian scores, reflecting how surprising each point is based on the prior and data likelihood.

5.3 Identify candidate change points

Once the posterior probabilities are computed, we extract meaningful change points from the sequence. This requires turning a continuous curve of posterior scores into a discrete list of specific time steps that are likely to represent actual transitions.

We first apply a fixed threshold τ (e.g., 0.5). All time steps where the posterior value exceeds this threshold are considered candidate change points. However, since BOCPD may produce multiple high posterior values around a true change, we add a filtering rule to avoid redundant detections.

We use a threshold of $\tau = 0.5$ to pick out candidate change points. This means we mark time steps where the posterior probability exceeds 50%. The idea is simple: if the probability that a time step is a change point is greater than half, we treat it as a reasonable candidate. This makes 0.5 a natural and intuitive starting rule. This is only the first filter, not the final decision.

We do not perform a full sensitivity analysis here, because in the next stage, our MCMC method will already explore the uncertainty and summarize the distribution of possible change points more thoroughly.

Specifically, we enforce a minimum distance d between detected change points. If two or more exceed the threshold within d time steps, we keep only the one with the highest posterior.

We formalize this as:

$$\text{ischange-point}(i) = \begin{cases} 1, & \text{if prob}[i] > \tau \text{ and } |i - \text{last}| > d \\ 0, & \text{otherwise} \end{cases}$$

Here, last refers to the index of the most recently accepted change point. The filtering rule ensures that the selected change points are both high confidence and temporally well separated.

6 Detection of Dominant Change-Point by BOCPD

Although the previous step extracts candidate change points, these points may still be ambiguous, sensitive to the threshold, or occur in clusters. This can make interpretation difficult, especially when we want to summarize the dynamics using a single representative transition.

To address this, we define a dominant change point based on the posterior distribution. Instead of relying on isolated peaks, we analyze the posterior over local regions and identify the segment where change is most strongly supported.

6.1 Window scan over the posterior

To identify the most supported region of change, we slide a fixed-size window across the posterior sequence and compute the total posterior mass within each window.

Let w be the window size, and let prob$[i]$ denote the posterior value at time i. For each center position c, we define a symmetric window $[c - \frac{w}{2},\ c + \frac{w}{2}]$, and compute the total posterior mass in that window:

$$S(c) = \sum_{i=c-w/2}^{c+w/2} \text{prob}[i]$$

We then select the center c^* that maximizes this quantity:

$$c^* = \arg\max_c S(c)$$

This gives us the region where the posterior evidence for change is highest. It smooths out local noise and gives preference to regions with sustained high posterior values, rather than isolated spikes.

6.2 Dominant change point selection

After identifying the region with the highest cumulative posterior mass, we define the dominant change point as the midpoint of this region.

Formally, if c^* is the center of the window that maximized the sum:

$$c^* = \arg\max_c \sum_{i=c-w/2}^{c+w/2} \text{prob}[i]$$

then the final dominant change point is simply:

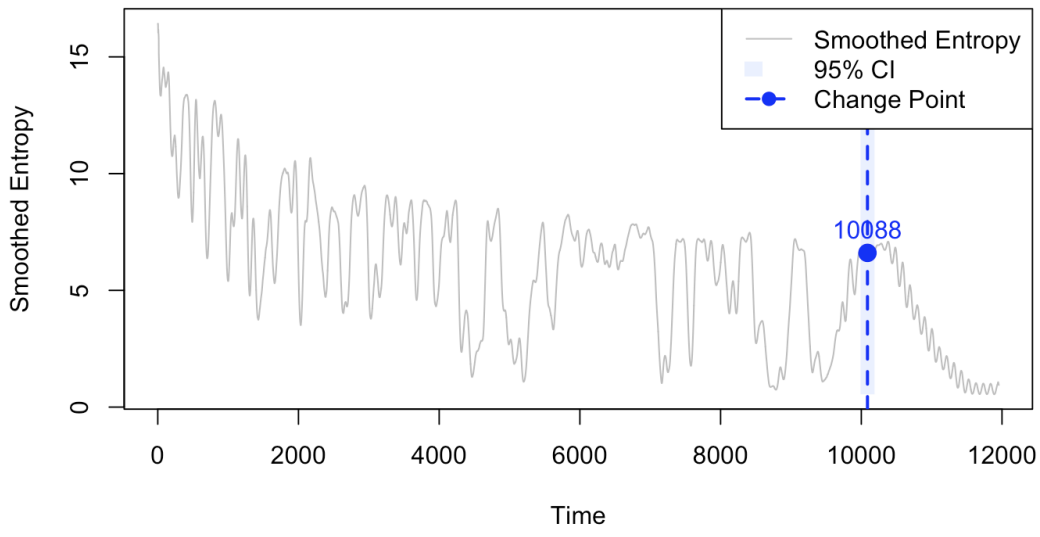
$$\text{change\_point} = c^*$$

This approach ensures that the selected change point is not only locally significant, but also represents a sustained shift supported by the posterior across a region. It addresses the issue of non-uniqueness and provides a consistent decision rule for extracting a single most probable transition.

6.3 Result Summary

After running the BOCPD method on the smoothed entropy sequence, we identified the dominant change point at index 10088. The associated 95% confidence interval for this change-point is [9989, 10188].

Figure 3: Dominant Change Region via BOCPD



7 General Method: MCMC Sampling

BOCPD gives us a fast and structured way to find change points, but it still depends on fixed assumptions like the hazard rate and assumes the data follows a normal distribution. These choices are not always easy to justify. In many cases, we want the model to learn directly from the data, instead of relying on tuning.

To make the method more flexible, we move toward a general Bayesian approach. This time, instead of computing a fixed posterior at each step, we use MCMC to sample from the full posterior over change point locations. Using MCMC sampling instead of a fixed posterior allows us to account for uncertainty and explore different possible configurations, not just the most likely one.

7.1 Modeling the change-point and Priors

In this section, we aim to identify a change-point in the smoothed entropy sequence $x = (x_1, x_2, \ldots, x_T)$, which is hypothesized to split the series into two segments with different means. Let $c \in \{5, \ldots, T-5\}$ denote the change-point. This restriction ensures that both segments contain sufficient data points for stable estimation, avoiding change-points that are too close to the sequence boundaries.

We assume that the observations before and after the change-point follow Normal distributions with different means but a common known variance σ^2. Specifically, we set:

$$x_t \sim \begin{cases} \mathcal{N}(\mu_1, \sigma^2), & \text{for } t \leq c \\ \mathcal{N}(\mu_2, \sigma^2), & \text{for } t > c \end{cases}$$

To reflect prior uncertainty, we assign the following priors:

$$c \sim \text{Uniform}(5, T-5), \quad \mu_1 \sim \mathcal{N}(7, 1.5^2), \quad \mu_2 \sim \mathcal{N}(3, 1^2)$$

The prior means of μ_1 and μ_2 are set based on average estimates obtained from the BOCPD posterior in the previous section. The prior variances are fixed to reasonable default values, allowing for moderate uncertainty in each segment's mean while ensuring regularization.

Then the unnormalized posterior distribution becomes:

$$\pi(c, \mu_1, \mu_2 \mid x) \propto p(x \mid c, \mu_1, \mu_2) \cdot p(c) \cdot p(\mu_1) \cdot p(\mu_2)$$

To improve numerical stability and simplify computation, we take the logarithm of the unnormalized posterior. This avoids numerical underflow from multiplying small likelihood values, preserves the distribution's mode, and transforms products into sums for easier computation. Thus, we work with the log-posterior:

$$\log \pi(c, \mu_1, \mu_2 \mid x) = \sum_{t=1}^{c} \log \mathcal{N}(x_t \mid \mu_1, \sigma^2) + \sum_{t=c+1}^{T} \log \mathcal{N}(x_t \mid \mu_2, \sigma^2) + \log \mathcal{N}(\mu_1 \mid 7, 1.5^2) + \log \mathcal{N}(\mu_2 \mid 3, 1^2)$$

This posterior serves as the target distribution from which we draw samples using the Metropolis-Hastings algorithm in the next section.

7.2 Metropolis-Hastings Sampling Algorithm

To sample from the posterior distribution of (c, μ_1, μ_2), we use the Metropolis-Hastings (MH) algorithm. This algorithm builds a Markov chain that eventually follows the target posterior distribution. At each step, we suggest a new candidate value and decide whether to accept it based on how likely it is compared to the current one.

Let $\theta = (c, \mu_1, \mu_2)$ be the current value. For each iteration t, the MH algorithm proceeds as follows:

1. Propose a new change-point using a random step:

$$c^{\text{new}} = c^{(t)} + \epsilon_c, \quad \epsilon_c \in \{-1, +1\}$$

2. Propose new mean values by adding random noise from a normal distribution:

$$\mu_1^{\text{new}} = \mu_1^{(t)} + \eta_1, \quad \mu_2^{\text{new}} = \mu_2^{(t)} + \eta_2$$

$$\eta_1, \eta_2 \sim \mathcal{N}(0, \sigma^2_{\text{prop}})$$

   Here, σ^2_{prop} is a fixed proposal variance controlling the step size.

3. Compute the log of the posterior for both the new and current values:

$$\log \pi^{\text{new}} = \log \pi(c^{\text{new}}, \mu_1^{\text{new}}, \mu_2^{\text{new}} \mid x)$$

$$\log \pi^{\text{old}} = \log \pi(c^{(t)}, \mu_1^{(t)}, \mu_2^{(t)} \mid x)$$

4. Accept the new value with probability:

$$\alpha = \min\left(1, \exp(\log \pi^{\text{new}} - \log \pi^{\text{old}})\right)$$

If the new value is accepted, we move to it. If not, we stay at the current value. This method lets us explore the full posterior even when it cannot be computed exactly.

7.3 Initial Setup and Burn-in Strategy

To ensure robust results and avoid getting stuck in local modes, we run four MCMC chains with different initial values. Each chain uses 100,000 iterations, which gives the sampler enough time to explore the full posterior distribution. We also set a proposal variance (`prop.var`) to control the step sizes when proposing new values for μ_1 and μ_2. This helps balance the trade-off between acceptance rate and movement through the parameter space.

The initial values for c, μ_1, and μ_2 are chosen within a reasonable range around the change-point and average values before and after the change detected by BOCPD. This improves the chance of reaching convergence.

To reduce the effect of initialization, we discard the first 20% of each chain as burn-in. These early samples might be more influenced by the starting point and not yet representative of the posterior. Removing them helps us focus on the stable part of the chain. We checked that this removal had little impact—the results before and after burn-in looked very similar, showing that convergence had already been achieved.

7.4 Convergence Diagnostics

We looked at the combined trace plots of μ_1 and μ_2 for all four chains. The lines show stable and dense fluctuations, looking like a "busy hedge." This suggests the chains have mixed well and likely reached convergence.

Figure 4: Trace plot of μ_1 across four chains

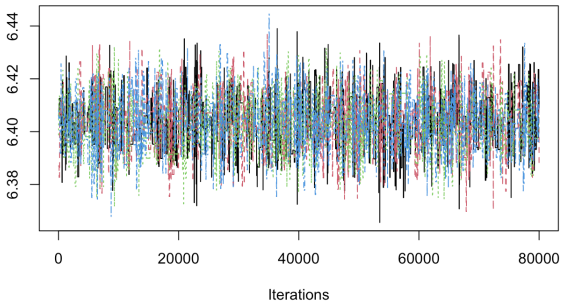
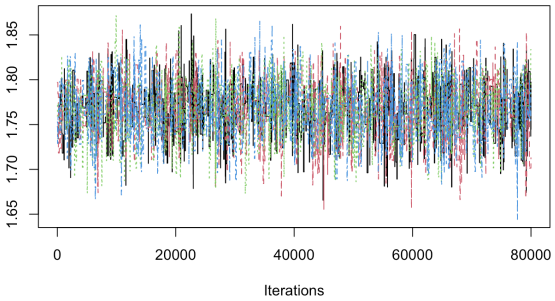


Figure 5: Trace plot of μ_2 across four chains



We also checked the trace plots for each chain separately. We included μ_1, μ_2, and the change-point c for all four chains. As shown in Figure 6 and Figure 7, each plot shows stable movement without any clear shift or unusual pattern, which supports the same conclusion.

We also applied the Gelman-Rubin diagnostic to further check convergence. GR compares the variance within chains to the variance between chains. If the chains have mixed well, these values should be similar.

In our case, the potential scale reduction factors (PSRFs) for μ_1, μ_2, and c are all very close to 1. The multivariate PSRF is also 1.01, which is below the common threshold of 1.1. This confirms that convergence has likely been reached.

Figure 8: Gelman-Rubin diagnostic values for each parameter

```
Potential scale reduction factors:

    Point est. Upper C.I.
c         1.01       1.01
mu1       1.00       1.00
mu2       1.00       1.01

Multivariate psrf

1
```

Besides, we further used the Raftery-Lewis diagnostic to evaluate how many iterations are needed to estimate the target quantile accurately. This method is especially helpful when assessing convergence for quantile-based estimates, such as posterior medians or credible intervals.

In our results, the Raftery-Lewis values are relatively large (see Figure 9) . This is not surprising, as our model involves a discrete change-point parameter that tends to stay constant for many iterations. This leads to strong autocorrelation in the Markov chains and inflates the number of effective samples required.

Although the values appear high, this behavior is typical and acceptable in change-point detection problems. Detailed results are provided in the Appendix.

7.5 Posterior Summary and change-point Estimation

After verifying convergence, we combined all four chains to estimate the posterior means and 95% credible intervals for each parameter. The summarized results are shown in table, providing a clear summary of both the location of the change-point and the level of change between segments. The estimated change-point is 10685.34, with a 95% credible interval ranging from 10127.00 to 10735.00.

Figure 10: Posterior summary of c, μ_1, and μ_2

	Mean	Lower Bound	Upper Bound
c	10686.25	10128.00	10736.00
μ_1	6.41	6.38	6.44
μ_2	1.87	1.71	3.22

8 Conclusion

In this study, we applied two Bayesian approaches—Bayesian Online change-point Detection (BOCPD) and a custom Metropolis-Hastings (MCMC) sampler—to detect change-points in the smoothed entropy sequence as early warning signals. Both methods identified significant shifts, but gave different change-point estimates: BOCPD at index 10088, and MCMC around index 10685.

While both are Bayesian, MCMC offers a more comprehensive and principled framework. Unlike BOCPD, which uses a local sliding window and fixed hazard rate, MCMC fully explores the parameter space, yielding posterior distributions, credible intervals, and diagnostic tools without relying on heuristic settings.

The later change-point from MCMC is reasonable—it reflects its ability to detect broader and more sustained shifts, compared to the earlier and more localized detection from BOCPD. Given this, we consider the MCMC estimate at index 10688 to be more reliable and scientifically grounded.

References

[1] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[2] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, 2017.

[3] R. Killick and I. A. Eckley, "changepoint: An r package for changepoint analysis," *Journal of Statistical Software*, vol. 58, no. 3, pp. 1–19, 2014.

[4] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[5] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*. CRC Press, 3 ed., 2013.

[6] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[7] D. Eddelbuettel and R. Francois, "Rcpp: Seamless r and c++ integration," *Journal of Statistical Software*, vol. 40, no. 8, pp. 1–18, 2011.

[8] S. Martínez and O. A. Rosso, "Complexity-entropy causality plane: A useful approach for distinguishing songs," *EPL (Europhysics Letters)*, vol. 89, no. 5, p. 58003, 2010.

[9] L. Zunino, M. C. Soriano, and O. A. Rosso, "Characterization of chaotic maps using the permutation entropy-complexity plane," *Physical Review E*, vol. 86, no. 4, p. 046210, 2012.

[10] G. Shmueli and J. Polak, *Practical Time Series Forecasting with R: A Hands-On Guide*. Axelrod Schnall Publishers, 3 ed., 2021.

Figure 6: Combined trace plots for μ_1 and μ_2 across four chains

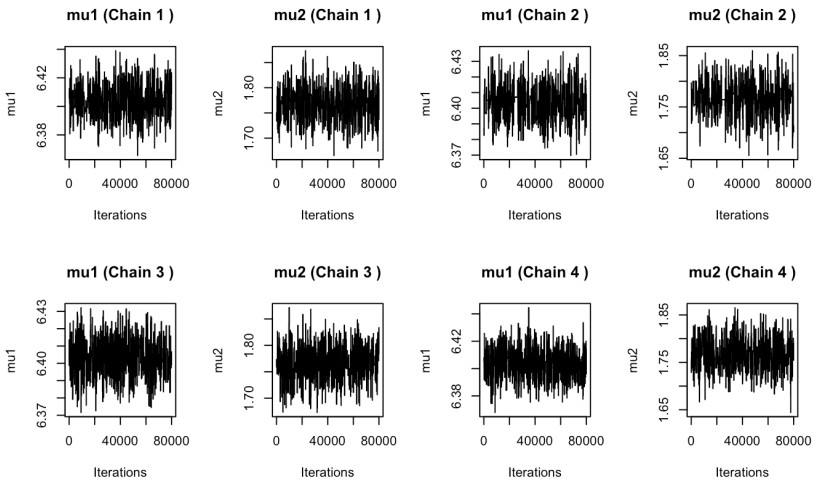


Figure 7: Combined trace plots for c across four chains

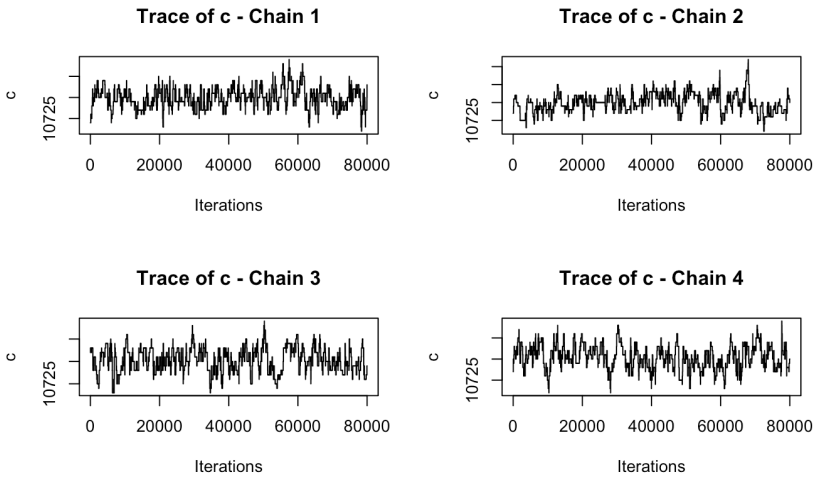


Figure 9: Combined trace plots for c across four chains

```
========================
Raftery Diagnostic - Chain 1
========================
        M      N Nmin     I
c   17336  17336 3746  4.63
mu1   199 216653 3746 57.80
mu2   312 344995 3746 92.10

========================
Raftery Diagnostic - Chain 2
========================
       M       N Nmin    I
c   8962 5273221 3746 1410
mu1  211  228655 3746   61
mu2  432  511224 3746  136

========================
Raftery Diagnostic - Chain 3
========================
        M      N Nmin      I
c   17398  17398 3746   4.64
mu1   180 193705 3746  51.70
mu2   382 418730 3746 112.00

========================
Raftery Diagnostic - Chain 4
========================
        M      N Nmin      I
c   17280  17280 3746   4.61
mu1   208 228422 3746  61.00
mu2   373 409340 3746 109.00
```