Monte Carlo and Empirical Methods for Stochastic Inference

Home Assignment 3 Solution

Alessandro Celoria
Ruoyi Zhao

March 2023

1 Coal Mine Disasters Intensity

We are given historical data on coal mine accidents during a more than 300 years long period spanning from 1658 to 1980. Our goal is to analyze this data to infer statistical evidence on the frequency, intensity, and trend of these events for the entire period. It is immediately clear that a very simple linear model is not going to be able to accurately fit the data as the immense changes that occurred in our society during this period significantly altered the incident rate in coal mines through the time window under scrutiny. This is what we refer to when we talk about *trend* and we are going to solve this problem through the concept of *breakpoints*, that is dividing the period into sub-periods and finding independent accident intensities $\lambda = (\lambda_1 \ \dots \ \lambda_n)$ for each of them to achieve a more accurate fit and relevant data.

The Data We model the time between accidents as an exponential random variable with intensity λ_i for the *i-th* period. Considering the *i-th* period to cover $[t_i \ t_{i+1})$ and calling λ_i and N_i the intensity and number of accidents for each period we can describe the time difference $\Delta\tau$ between subsequent accidents in the *i-th* interval as a random variable distributed as follows:

$$\Delta\tau \ \sim \ Exp(x;\lambda_i) = \lambda_i e^{-\lambda_i x}$$

Furthermore we have that $\lambda_i \sim \Gamma(2,\theta)$ and in turn $\theta \sim \Gamma(2,\Psi)$. So we have both a prior and *hyper*-prior to λ, with Ψ being a hyperparameter of our choice.

Breakpoint Distribution The distribution of breakpoints is crucial to a good fit to the data, and in order to guarantee successful statistical inference we are not going to set fixed breakpoints but rather choose a number of breakpoints and evolve their position over time through a particle filter. Since we don't have a prior distribution for the breakpoints we need to choose one ourselves, and to do so we can exploit some basic assumptions and goals about the shape and general properties of a suitable breakpoint distribution:

- Breakpoints **must** be ordered.
- Breakpoints should not be too close as that would defeat their point.
- The first and last breakpoints **must** coincide with the start and end of the interval.

These concepts can be synthesized in the following distribution:

$$f(t) \propto \begin{cases} \prod_{i=1}^{d}(t_{i+1}-t_i) & \text{for } t_1 < t_2 < \cdots < t_d < t_{d-1} \\ 0 & \text{otherwise} \end{cases}$$

All of the above implies the following distribution for τ:

$$f(\tau|\lambda,t) = e^{-\sum_{i=1}^{d}\lambda_i(t_{i+1}-t_i)}\prod_{i=1}^{d}\lambda_i^{n_i(\tau)}$$

The Method We want to sample from the posterior $f(\theta, \lambda, t|\tau)$ to infer the distribution of the accidents from the historical data we observed, however we do not have access to this posterior and cannot write down its formula, therefore to solve this problem we have to break up the posterior distribution in three blocks and update them independently using **Gibbs sampling**. The blocks correspond to the individual marginals of the a-posteriori variables:

- $f(\theta|\lambda, t, \tau)$
- $f(\lambda|\theta, t, \tau)$
- $f(t|\theta, \lambda, \tau)$

1.1 Computing the marginals

We are now interested in computing and identifying the marginals of these three variables in order to be able to sample from them in the update step of the Gibbs sampler. Since we only use these for sampling purposes we are only interested in preserving the shape of the distribution and can thus ignore any normalization constant we may not be able to include in the formula of a well-known distribution, since our priority is to have a distribution from which we can easily sample.

The θ Marginal We start with the marginal for theta and use Bayes' theorem in order to decompose it into a product of known priors and posteriors:

$$f(\theta|\lambda, t, \tau) = \frac{f(\theta, \lambda, t, \tau)}{f(\lambda, t, \tau)} = \frac{f(\tau|\lambda, t, \theta)f(t)f(\lambda|\theta)f(\theta)}{f(\tau|\lambda, t)f(t)f(\lambda)} = \frac{f(\tau|\lambda, t, \theta)f(\lambda|\theta)f(\theta)}{f(\tau|\lambda, t)f(\lambda)}$$

We know all distributions showcased above except for $f(\tau|\lambda, t, \theta)$. To get rid of this inconvenient distribution we have to consider the Bayesian network of the variables involved in this particular instance, which is shown in Figure 1: as we can see θ forms a chain through λ to reach τ, and Bayesian theory tells us that by conditioning on λ we make the two independent. Therefore $f(\tau|\lambda, t, \theta) \equiv f(\tau|\lambda, t)$ and with this simplification we can write:

$$f(\theta|\lambda, t, \tau) = \frac{f(\tau|\lambda, t)f(\lambda|\theta)f(\theta)}{f(\tau|\lambda, t)f(\lambda)} = \frac{f(\lambda|\theta)f(\theta)}{f(\lambda)} \propto f(\lambda|\theta)f(\theta)$$

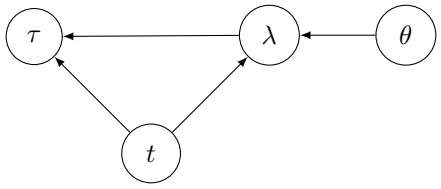


Figure 1: Bayesian network for $f(\tau|\lambda, t, \theta)$

To identify this distribution we can consider that we have a prior $f(\theta) = \Gamma(2, \Psi)$ and a likelihood $f(\lambda|\theta) = \Gamma(2, \theta)$ which results in a posterior $f(\theta|\lambda) \propto f(\lambda|\theta)f(\theta)$. If we can then find the posterior we can find the (non-normalized version of the) marginal distribution for θ. Luckily the table found at *L11 - page 16* conveniently tells us that the distribution we seek is $f(\theta|\lambda, t, \tau) = \Gamma(2 + 2d, \Psi + \sum_{i=1}^{d} \lambda_i)$.

The λ Marginal We follow a similar process for the distribution of λ:

$$f(\lambda|\theta, t, \tau) = \frac{f(\lambda, \theta, t, \tau)}{f(\theta, t, \tau)} = \frac{f(\tau|\lambda, t, \theta)f(\lambda|\theta)f(\theta|t)f(t)}{f(\theta, t, \tau)} = \frac{f(\tau|\lambda, t)f(\lambda|\theta)f(\theta)f(t)}{f(\theta, t, \tau)} \propto f(\tau|\lambda, t)f(\lambda|\theta)f(\theta)f(t)$$

This time we exploited the *chain rule* to rewrite $f(\lambda|\theta, t)f(\theta, t)$ as $f(\lambda|\theta)f(\theta|t)f(t)$ and then the fact that θ is independent of t to solve the problem. This time there is no *clear* likelihood and posterior we can take from

the table so we resort to some manual computations. We consider the distribution for a single component λ_i of the λ vector, and of course, such a component is only going to be influenced by the breakpoints for its own interval, which is t_{i+1} and t_i. With this in mind, we write:

$$\begin{aligned} f(\lambda_i|\theta, t_{\lambda_i}, \tau) &= f(\tau|\lambda_i, t_{\lambda_i}) f(\lambda_i|\theta) f(t_{\lambda_i}) = \lambda_i^{n_i(\tau)} e^{-\lambda_i(t_{i+1}-t_i)} \cdot \Gamma(2,\theta) \cdot \Gamma(2,\Psi) \cdot (t_{i+1}-t_i) = \\ &= \lambda_i^{1+n_i(\tau)} e^{-\lambda_i(\theta+t_{i+1}-t_i)} (t_{i+1}-t_i) e^{-\Psi\theta} \theta^2 \Psi^2 = \\ &= \Gamma(n_i(\tau)+2, \theta+t_{i+1}-t_i) \frac{\theta^2\Psi^2(t_{i+1}-t_i)(n_i(\tau)+2)! e^{-\Psi\theta}}{(\theta+t_{i+1}-t_i))^{n_i(\tau)+2}} \\ &\propto \Gamma(n_i(\tau)+2, \theta+t_{i+1}-t_i) \end{aligned}$$

The t Marginal Finally we do the same for the distribution of t:

$$f(t|\theta,\lambda,\tau) = f(t|\lambda,\tau) = \frac{f(t,\lambda,\tau)}{f(\lambda,\tau)} = \frac{f(\tau|\lambda,t)f(\lambda)f(t)}{f(\tau|\lambda)f(\lambda)} = \frac{f(\tau|\lambda,t)f(t)}{f(\tau|\lambda)} \propto f(\tau|\lambda,t)f(t)$$

Again for this computation, we need some manual computations. We compute the marginals for time t_i and the corresponding intensity λ_i:

$$f(t_i|\theta,\lambda_i,\tau) \propto f(\tau|\lambda_i,t)f(t_i) = \lambda_i^{n_i(\tau)}(t_{i+1}-t_i)e^{-(t_{i+1}-t_i)\lambda_i} = \lambda_i^{n_i(\tau)-2}\Gamma((t_{i+1}-t_i);1,\lambda_i) \propto \Gamma((t_{i+1}-t_i);1,\lambda_i)$$

As we can see the resulting distribution is a distribution for the *length* of the intervals and **not** for the breakpoints themselves, however, this is not a problem since the endpoints of the interval are fixed and we can easily convert between lengths and breakpoints knowing $t_1 = 1658$.

The problems Unfortunately the distribution of t causes us some problems: first of all, since we are sampling lengths it means that we are obtaining the length of the final interval from a stochastic distribution as well, but as previously stated the endpoints are fixed, so this would require a truncated distribution, which requires handling of special cases. This makes sampling from the marginal difficult. To this extent, the decision has been taken to resort to a **mixed sampler** where every variable but t gets sampled through the *Gibbs* method but t is sampled through the *Metropolis-Hastings* method using a random walk proposal.

1.2 Updating the Breakpoints

We are given two options for the random walk proposal:

- **Update all Breakpoints at once:** compute a mutation for every breakpoint and then sample it
- **Update breakpoints individually:** compute and sample a mutation individually for every breakpoint

We opted for the second choice as we felt it would lead to better mixing as the chances of no breakpoints being updated at a given iteration would be significantly lower than with the first proposal. Both proposals are random walk proposals, which is a class of symmetric proposals. This means that we can simplify the acceptance probability formula to:

$$\alpha(t_i^k, t_i k+1) = \min\left(1, \frac{f(t_i^{k+1}|\lambda_i,\theta,\tau)}{f(t_i^k|\lambda_i,\theta,\tau)}\right)$$

The fact taht the *one-at-a-time* random walk proposal kernel is symmetrical also means that the chain is aperiodic and irreducible, satisfying the requirements for a valid proposal kernel for the HM algorithm. For each breakpoint update we mutate it as $t_i^* \leftarrow t_i + \epsilon$ where $\epsilon \sim U(-R,R)$ and $R = \rho(t_{i+1}-t_{i-1})$ with ρ being another hyperparameter. We finally have enough information to build our MCMC algorithm.

1.3 The MCMC Estimator

We write a simple algorithm to estimate the posterior $f(\theta, \lambda, t|\tau)$ by using Markov Chain Monte Carlo with a hybrid approach that uses Gibbs Sampling to update λ and θ and the Metropolis-Hastings sampler to update the breakpoints t. The algorithms used are reported in Algorithm 1 and 2.

Algorithm 1 MCMC Algorithm

$\theta_0 \sim \Gamma(2, \Psi)$
$\lambda_0 \sim \Gamma(2, \theta)$

for $k \leftarrow 1 \dots N$ **do**
    $\theta_k \sim f(\theta_k|\lambda_{k-1}, t_{k-1}, \tau)$
    $\lambda_k \sim f(\lambda_k|\theta_k, t_{k-1}, \tau)$
    $t \leftarrow \text{mh}(\lambda_k, \theta_k)$
end for

Algorithm 2 MH Algorithm

$t^k \leftarrow t^{k-1}$

for $n \leftarrow 1 \dots i$ **do**
    $t_i^k \sim r(t_i^k|t^{k-1})$
    $\alpha \leftarrow \min\left(1, \frac{f(t^k|\lambda_k, \theta_k, \tau)}{f(t^{k-1}|\lambda_k, \theta_k, \tau)}\right)$

    if $\alpha > \mathcal{U} \sim U(0, 1)$ **then**
        $t_i^{k+1} \leftarrow t_i^k$ ▷ Reject the change with probability $1 - \alpha$
    end if
end for

1.4 Hyperparameters

We are now left to determine the desired values of Ψ and ρ. To do so we had to choose a number of breakpoints we deemed appropriate to present as a final result for this problem and perform a grid search to understand how these hyperparameters affect the algorithm and choose values that could lead us to a satisfactory result. We opted for 5 breakpoints, thus $d = 6$. In Figure 2 we plot the variance of θ and the mean-variance of λ and t for every tested combination of breakpoints.

Effect of Ψ As we can see Ψ does not have a significant effect on either λ or t but is the main contributor to changes in the variance of θ, which is not surprising since $f(\theta) = \Gamma(2, \Psi)$.

Effect of ρ We found negligible effect of ρ on θ (again, not surprising) but as expected ρ had a deep influence on both λ and especially on t. We can see how the variance increases dramatically for lower values of ρ. What is not visible from this figure is that for the lowest tested value ($\rho = 0.001$) the variance is quite low. This can be easily explained as this hyperparameter effectively acts like a "learning rate" of sorts for the algorithm, since it has the ability to greatly restrict (or not) the mixing rate of t and thus how quickly the breakpoints are allowed to update and find their optimal position. With a very restrictive ρ the breakpoints are only allowed to change by a very tiny margin each iteration, making the algorithm very stable (low variance) but also not allowing convergence to the optimal result in a useful time. We then decided to ignore this edge case and concentrate on the rest of the parametric grid.

Optimal parameters We wanted to find a combination (Ψ, ρ) that would minimize the variance for both λ and t, whereas the variance of θ was not interesting to us. We prioritized the variance of λ since it's

ultimately what we aim at estimating and it has a strong dependence on t anyways, furthermore t would naturally be more susceptible in variance to changes to ρ and we expected a higher degree of variance for that variable anyways. After some evaluation, we settled on values $\Psi = 8$ and $\rho = 0.05$. This value of ρ is just at the "feet of the mountains" in the variance plot for λ and a bit higher on the "side of the mountains" in the variance plot for t. After analyzing the estimation performance with these parameters we concluded that while t does express a significant variance, it does not "jump around" as much as with other values after the initial *burn-in* period and we were satisfied with the results.

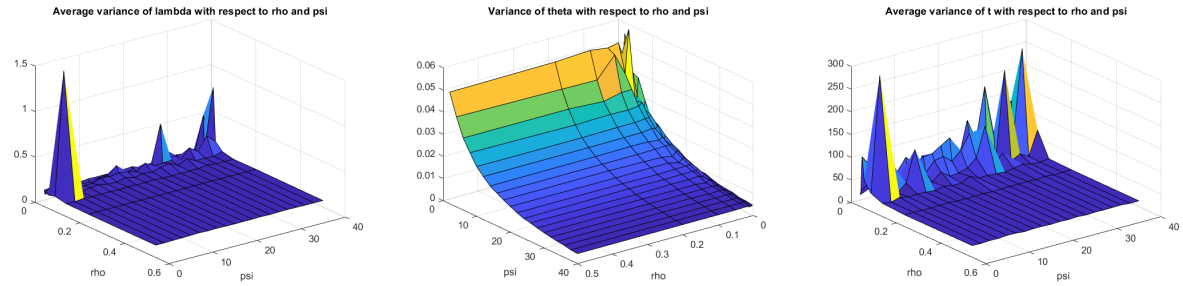


Figure 2: Variances for all three posterior variables with respect to Ψ and ρ

Ψ	2	4	6	8	10	12	14	16	18	20
	22	24	26	28	30	32	34	36	38	40
ρ	0.002	0.005	0.01	0.02	0.05	0.1	0.2	0.5		

Table 1: Set of hyperparameters tested during the grid search.

Acceptance Rate We plot the acceptance rate of new breakpoints with respect to the same values of ρ listed in Table 1. We can see the results in Figure 3. We can see how the acceptance rate declines steeply for higher values of ρ. This is expected as a bigger ρ means that the steps of the random walk can get larger and breakpoints can thus more easily get close to each other. Since the prior $f(t)$ highly depends on the distance between breakpoints, with higher likelihoods being associated with more distance between points, having a large value for ρ can more easily lead to the rejection of a breakpoint update.

Mixing The highest recorded acceptance rate (for $\rho = 0.001$) is $\approx 81\%$ whereas the lowest (for the absurd value of $\rho = 0.5$) is $\approx 4.7\%$. We can notice how the choice of ρ has a huge effect on the acceptance rate, as expected. We know that the rule of thumb for a good mixing is to have an acceptance rate of $\approx 30\%$. With $\rho = 0.05$, which is what we previously observed to work best for the performance of this algorithm, we obtain an acceptance rate of 29.08% which is very close to 30% and indeed the closest we get with the parameters we explored. This confirms and validates our previous intuition on the ideality of this parameter as we can see that other choices tend to either forget too fast ($\rho < 0.05$) or too slow ($\rho > 0.05$), with the former being more egregious at that.

1.5 How many breakpoints?

In Section 3.1 we compare the fitness and historical behavior of the breakpoint distribution for a number of breakpoints between 1 and 6 with the hyperparameter values we found above. We also compare the variance figures for each result and display them in 4. As we can see for $d = 2$ and $d = 3$ (1 and 2 breakpoints respectively) the breakpoints are very stable but the fit to the curve is not very good. For $d = 4$, $d = 5$, and $d = 6$ we have increasingly more accurate fits to the curve (with $d = 6$ being almost perfectly overlaid over the data), but with more variance (which mainly shows as increased jitter since the "jumpyness" of the function does not seem affected). With $d = 7$ everything breaks down as there are some intervals that hold too few accidents to make a stable and accurate prediction of the λ_i for that interval. We also see

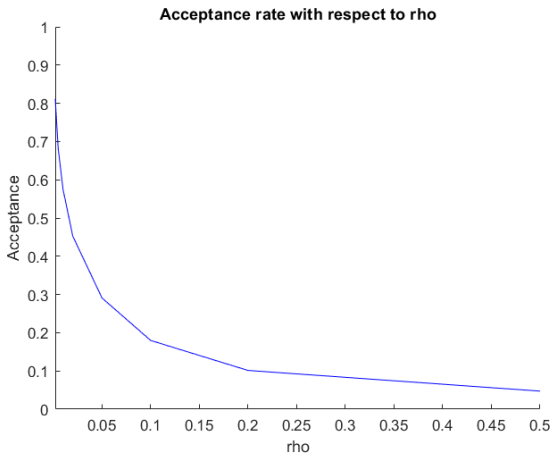


Figure 3: Acceptance rate for new breakpoints with respect to different values of ρ

this in the fit: despite having more flexibility the curve does not fit the data as much, due to the erratic behavior seen well past the end of the *burn-in* period. We can notice how the breakpoints seem to stabilize later on, which suggests that an increased number of iterations in the *burn-in* might lead to better results, however the increased unreliability of the algorithm we observed during experimentation with this high count of breakpoints discouraged us from pursuing further results for $d = 7$.

2 Parametric Bootstrap for the 100-year Atlantic Wave

Find the Inverse $F^{-1}(u; \mu, \beta)$ In this part, a *Gumbel Distribution* function is provided as:

$$F(x; \mu, \beta) = exp(-exp(-\frac{x-\mu}{\beta})), x \in \mathbf{R}$$

To find the inverse $F^{-1}(u; \mu, \beta)$, first, based on the above function we have:

$$F(F^{-1}(u)) = u$$

$$\Rightarrow exp(-exp(-\frac{F^{-1}(u; \mu, \beta) - \mu}{\beta})) = u$$

To obtain the inverse of this *Gumbel Distribution*, we take the logarithm of the equation, eliminating the exponentiation:

$$ln(-ln(u)) = -\frac{F^{-1}(u; \mu, \beta) - \mu}{\beta}$$

After standardizing the writing, the inverse:

$$F^{-1}(u; \mu, \beta) = \mu - \beta ln(-ln(u)$$

95% Confidence Intervals In the parametric bootstrap, the main idea is to assume the data from known distributions with unknown parameters. Therefore, in this task, the parameters (μ, β) are able to estimate by the provided function *est_gumbel.m* and refer to the code on page 21 of lecture 14 to obtain a 95% estimated confidence interval for parameters.
We generate new bootstrapped samples Y_b^*, $b \in 1, 2, ...B$ by the inverse *Gumbel Distribution* function we obtained from the previous task. And through these 1000 random samples, we perform the estimations of β, μ and compare them with the parameters' values we computed from the true data:

$$\Delta_b^* = \hat{\beta}(Y_b^*) - \hat{\beta}$$

$$\Delta_b^* = \hat{\mu}(Y_b^*) - \hat{\mu}$$

Besides, the estimated values of μ, β from the true data are:

$$\hat{\beta} = 1.485839630967733$$

$$\hat{\mu} = 4.147704597207478$$

Then sort the calculated difference, form the 95% confidence intervals With these two error terms, and obtain the results shown in the following table.

Parameters	Lower Bound	Upper Bound	Width Bound
β	1.4822	1.4896	0.0074
μ	4.1426	4.1532	0.0106

Table 2: The parametric bootstrapped 95% confidence intervals for $\beta\mu$.

One-sided 95% Confidence Intervals Based on the previous task, the goal is to perform a one-sided upper bound parametrically bootstrapped 95% confidence interval for the 100 years return value.
First, we have the T_th return value is:

$$F^{-1}(1 - \frac{1}{T}; \mu, \beta)$$

Where the total amount of observations $T = 3 \cdot 14 \cdot 100$ during 100-year.
Similarly to the above, we also use the inverse *Gumbel Distribution* function to compute the wave height for 1000 samples by parameter estimates from the previous task, the $\hat{\beta}$ and the $\hat{\beta}(Y_b^*)$ respectively. Afterward, we sort the differences between the two obtained wave height values:

$$sort(\hat{wave}(Y_b^*) - \hat{wave})$$

Then perform the upper bound of the 95% confidence interval

Parameter	Upper Bound	Estimation
wave_height	16.5720	16.5436

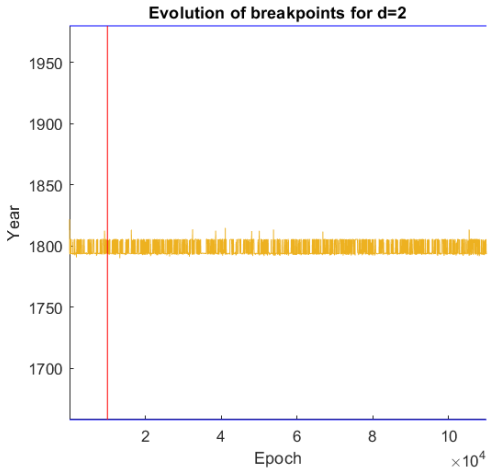
Table 3: The one-sided parametric bootstrapped 95% confidence intervals for the 100 years return value.

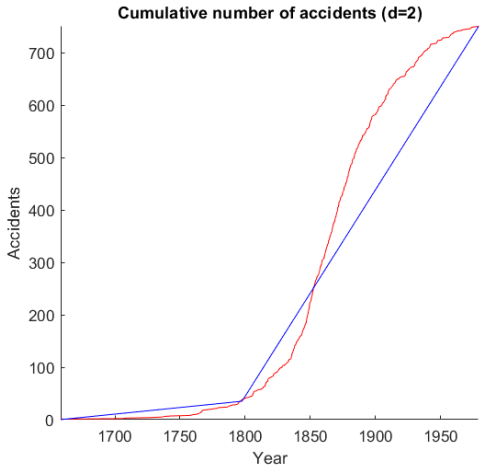
3 Appendix

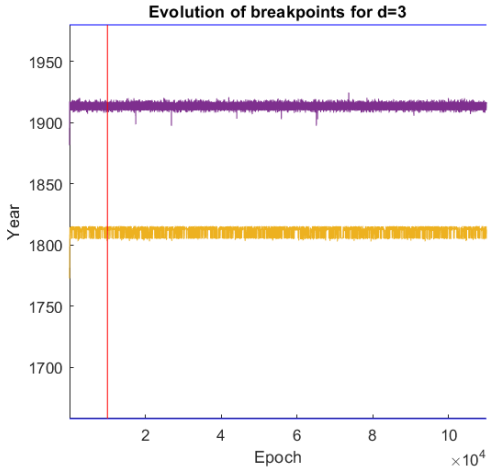
3.1 Behavior of the chain with a different number of breakpoints

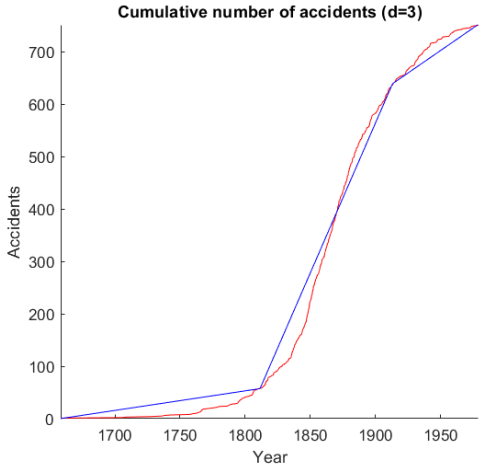
$\|t\|$	1	2	3	4	5	6
$\mathbf{V}\{\lambda\}$	1.563200e-02	3.746204e-02	9.879614e-02	7.386181e-02	7.136679e-02	1.224515e+00
$\mathbf{V}\{t\}$	9.569437e+00	5.046629e+00	5.873774e+01	1.788592e+01	1.283010e+01	7.806085e+01
$\mathbf{V}\{\theta\}$	4.077557e-02	3.210659e-02	2.956507e-02	3.136710e-02	2.725400e-02	2.365442e-02

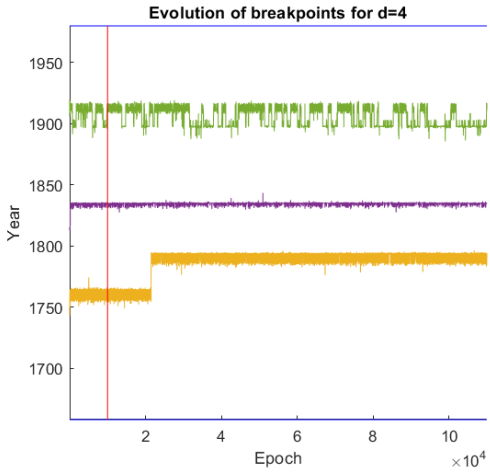
Table 4: Variance for all posteriors for each breakpoint count

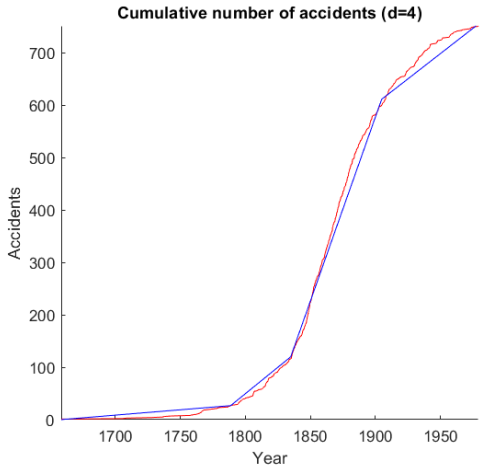

Evolution of breakpoints for d=2
Year
Epoch
×10^4

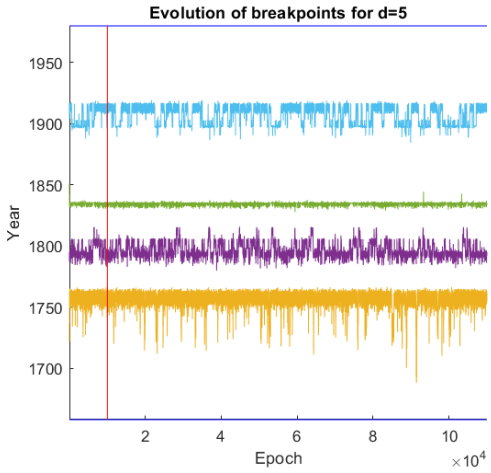

Cumulative number of accidents (d=2)
Accidents
Year

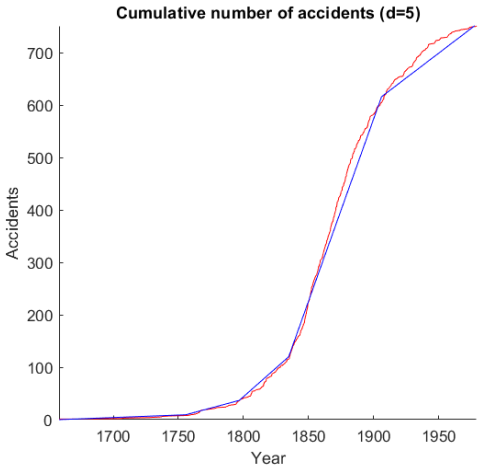

Evolution of breakpoints for d=3
Year
Epoch
×10^4

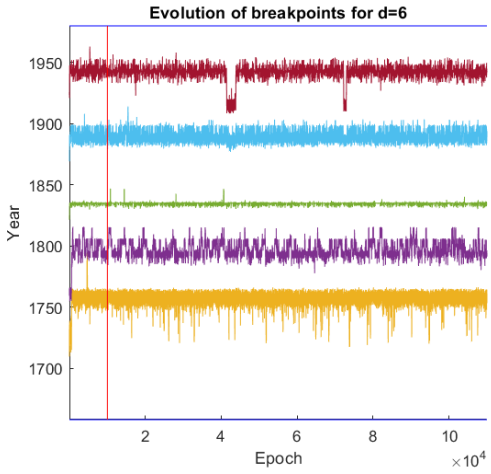

Cumulative number of accidents (d=3)
Accidents
Year

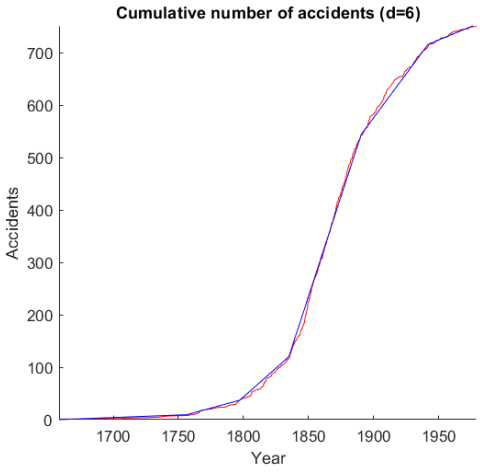

Evolution of breakpoints for d=4
Year
Epoch
×10^4

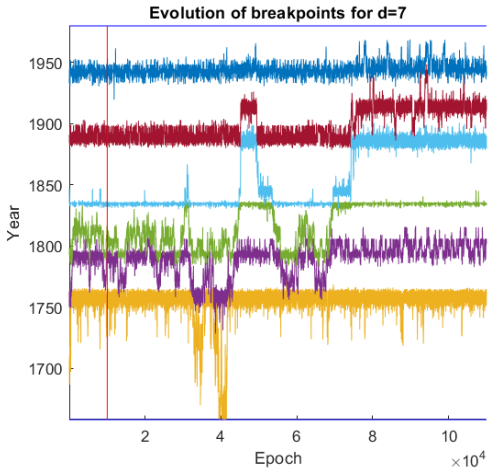

Cumulative number of accidents (d=4)
Accidents
Year

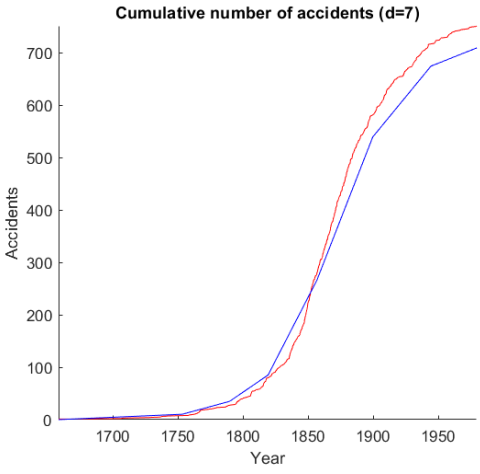

Evolution of breakpoints for d=5
Year
Epoch
×10^4


Cumulative number of accidents (d=5)
Accidents
Year


Evolution of breakpoints for d=6
Year
Epoch
×10^4


Cumulative number of accidents (d=6)
Accidents
Year


Evolution of breakpoints for d=7
Year
Epoch
×10^4


Cumulative number of accidents (d=7)
Accidents
Year