STAT 447 Assignment 9

MCMC Hacking

Caden Hewlett

2024-04-05

Data Import

The data we analyze records, for each of 74 days, the number of text messages sent and received by Davidson-Pilon.

```
# main data vector
sms_data = c(
    13,24,8,24,7,35,14,11,15,11,22,22,11,57,11,19,29,6,19,
    12,22,12,18,72,32,9,7,13,19,23,27,20,6,17,13,10,14,6,
    16,15,7,2,15,15,19,70,49,7,53,22,21,31,19,11,18,20,12,
    35,17,23,17,4,2,31,30,13,27,0,39,37,5,14,13,22)
# data frame
df = data.frame(
  num_texts = sms_data,
  day = 1:length(sms_data)
)
```

Bayesian Model

We denote C to be the change point, selected uniformly from days $d \in \{1, 2, \dots N\}$ where N is the number of observations. Then, there is a likelihood for days less than change point C and a likelihood for days above the change point.

We can denote the model as follows:

$$\begin{aligned} \lambda_1 &\sim \exp(1/100) \\ \lambda_2 &\sim \exp(1/100) \\ C &\sim \text{unif}\big(\{1, 2, \dots, N\}\big) \\ Y_d \mid C, \{\lambda_1, \lambda_2\} &\sim \text{pois}\big(\mathbb{1}[d < C]\lambda_1 + \mathbb{1}[d \geq C]\lambda_2\big) \end{aligned}$$

We will also refer to the $\{\lambda_1, \lambda_2\}$ pair as $\vec{\lambda}$.

We provide an implementation of the joint distribution of this model below.

```
# inputs are lambdas, C and y
log_joint = function(rates, change_point, y) {

  # Return log(0.0) if parameters are outside of the support
  if (rates[[1]] < 0 | rates[[2]] < 0 | change_point < 1 | change_point > length(y))
```

```
    return(-Inf)

  log_prior =
    dexp(rates[[1]], 1/100, log = TRUE) +
    dexp(rates[[2]], 1/100, log = TRUE)

  log_likelihood = 0.0
  for (i in 1:length(y)) {
    rate = if (i < change_point) rates[[1]] else rates[[2]]
    log_likelihood = log_likelihood + dpois(y[[i]], rate, log = TRUE)
  }

  return(log_prior + log_likelihood)
}
# log_joint(c(3.789566, 51.226584), 10, c(51, 54, 64, 49, 61))
```

Question 1: A Custom MCMC Sampler

NOTE We will briefly justify the reasoning for irreducibility of each kernel (and thus the combination of kernels) in Part 1 and leave invariance for the next section. Further, when we define these kernels, we will be doing so in terms of likelihoods γ. However, for more precision with respect to the true implementation, we will prove π-invariance with respect to the log joint model defined above.

Part 1: Algorithm

We will begin by defining two separate kernels K_1 and K_2 for the rate parameters $\{\lambda_1, \lambda_2\}$ and the change-point parameter C, respectively. We then unify these kernels by defining a kernel mixture for a selection probability ρ.

Let $C^\star$ and C be the proposed and current cutoff times, respectively. Similarly, we let $\vec{\lambda}^\star$ and $\vec{\lambda}$ be the proposed and current rates, respectively. We use $\vec{\lambda} = \{\lambda_1, \lambda_2\}$ and $\vec{\lambda}^\star = \{\lambda_1^\star, \lambda_2^\star\}$ interchangeably. Further, in each case, we define the proposal function using the following notation:

$$q\Big(\{\vec{\lambda}^\star, C^\star\} \mid \{\vec{\lambda}, C\}\Big) \equiv q\Big(\{\lambda_1^\star, \lambda_2^\star, C^\star\} \mid \{\lambda_1, \lambda_2, C\}\Big)$$

While slightly obtuse, this is intended to reflect the fact that the `rates` input is a vector, and the likelihood defined in the original model relies upon all three parameters. Further, this style allows us to maintain the same general form for q regardless of the kernel, which we can use to expedite the proof in Question 2.

We begin by discussing K_1. This kernel will only modify the $\vec{\lambda}$ parameter. Since each Poisson rate parameter is a real number, we will use a standard normal proposal.

Notably, rather than having a dimension of 2, we have each rate operate separately on their own means to be updated over time. The objective is to have more independent exploration for pre-and-post change point rates. Notably, our proposal vector is $\{\vec{\lambda}^\star, C\}$, highlighting that in this kernel we do not select new values for the change point.

$$\begin{aligned}
q_1\big(\{\vec{\lambda}^\star, C\} \mid \{\vec{\lambda}, C\}\big) &= \Big\{\frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}(\lambda_i^\star - \lambda_i)^2\Big) : i \in \{1, 2\}\Big\} \\
\alpha_1\big(\vec{\lambda}^\star, \vec{\lambda}, C\big) &= \min\Big\{1, r\big(\vec{\lambda}^\star, \vec{\lambda}, C\big)\Big\}, \text{ where } r\big(\vec{\lambda}^\star, \vec{\lambda}, C\big) = \frac{\gamma(\lambda^\star, C)}{\gamma(\lambda, C)} \\
K_1\big(\{\vec{\lambda}^\star, C\} \mid \{\vec{\lambda}, C\}\big) &= q_1\big(\{\vec{\lambda}^\star, C\} \mid \{\vec{\lambda}, C\}\big) \cdot \alpha_1\big(\vec{\lambda}^\star, \vec{\lambda}, C\big)
\end{aligned}$$

In a simplified version, we can write the equation below to simplify K_1 :

$$K_1\big(\vec{\lambda^\star} \mid \vec{\lambda}\big) = q\big(\vec{\lambda^\star} \mid \vec{\lambda}\big)\alpha\big(\vec{\lambda^\star}, \vec{\lambda}\big)$$

Further, we note that the standard normal proposal will allow the algorithm to explore all candidate $\vec{\lambda}$ pairs in $(\mathbb{R}^+)^2$. Let's talk briefly about why by highlighting an arbitrary proposed $\lambda_i^\star \in \vec{\lambda^\star}$ in terms of Poisson probability mass $\gamma(\vec{\lambda^\star}, C)$ for well-defined $C \in [1, N] \subseteq \mathbb{N}$.

Firstly, we note that Poisson rate parameters λ are strictly positive, while the standard normal surrounding λ_i can propose $\lambda_i^\star \in \mathbb{R}$. Notably, however, a property of the Poisson probability density (specifically for the `log_joint` function) is that:

$$p_{\text{pois}}(k;\lambda) = \mathbb{P}\big(\text{pois}(\lambda) = k\big) = \begin{cases} \dfrac{\lambda^k e^{-\lambda}}{k!}, & k \geq 0,\ \lambda \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Notably, in the case that <u>either</u> $\lambda_1^\star$ or $\lambda_2^\star$ are negative, the function will return zero. In this case the Metropolis-Hastings ratio will return zero, and the proposed $\vec{\lambda^\star}$ will be rejected. Hence, we know that in K_1 that not only will solely valid $\vec{\lambda}$ be explored, but further assuming that C is well-defined all candidate lambda pairs have nonzero probability mass; hence, all possible $\vec{\lambda^\star} \in (\mathbb{R}^+)^2$ can be explored from any given $\vec{\lambda}$ - thus this component of the kernel is irreducible. In the next section, we will verify the invariance of the overall kernel.

Now, we consider K_C. Ideally, we would like to define a symmetric proposal q_C; however, we cannot use a Normal Distribution as before so we must find a symmetric discrete distribution. As per the recommendation, this distribution should have variance greater than 3 to avoid slow mixing.

The simplest possible proposal in this situation would be the discrete $\text{unif}\big(\{a = 1, b = N\}\big)$ distribution, i.e.:

$$q\Big(\big\{\vec{\lambda}, C^\star\big\} \mid \big\{\vec{\lambda}, C\big\}\Big) = \frac{1}{b - a + 1} = \frac{1}{N - 1 + 1} = \frac{1}{N}$$

Where, in this case, $N = 74$. This choice ensures that every possible change point $C^\star$ can be proposed with nonzero probability given any current C, preserving irreducibility.

We can verify by computing the variance of this distribution (to check it will mix nicely):

$$\text{var}\Big(\text{unif}\,\{a, b\} \mid a = 1, b = N\Big) = \frac{(b - a + 1)^2 - 1}{12} = \frac{(N - 1 + 1)^2 - 1}{12} = \frac{N^2 - 1}{12} \approx 456 \text{ for } N = 74$$

Which is certainly a large enough variance to have a breadth of proposal options.

Then, we define the kernel K_2 in a similar fashion to before.

$$\begin{aligned} q_2\big(\{\vec{\lambda}, C^\star\} \mid \{\vec{\lambda}, C\}\big) &= \frac{1}{N} \\ \alpha_2\big(\vec{\lambda}, C, C^\star\big) &= \min\Big\{1, r\big(\vec{\lambda}, C, C^\star\big)\Big\}, \text{ where } r\big(\vec{\lambda}, C, C^\star\big) = \frac{\gamma(\lambda, C^\star)}{\gamma(\lambda, C)} \\ K_2\big(\{\vec{\lambda}, C^\star\} \mid \{\vec{\lambda}, C\}\big) &= q_2\big(\{\vec{\lambda}, C^\star\} \mid \{\vec{\lambda}, C\}\big) \cdot \alpha_2\big(\vec{\lambda}, C, C^\star\big) \end{aligned}$$

Finally, with each kernel defined above we can define the unified kernel K as follows. This is what we will be implementing in Part 3.

We use the definitions of K_1 and K_2 detailed above.

Let $\rho \sim \text{bern}(1/2)$

$$K\big(\{\vec{\lambda^\star}, C^\star\} \mid \{\vec{\lambda}, C\}\big) = \mathbb{1}[\rho = 1]K_1\big(\{\vec{\lambda^\star}, C\} \mid \{\vec{\lambda}, C\}\big) + \big(1 - \mathbb{1}[\rho = 1]\big)K_2\big(\{\vec{\lambda}, C^\star\} \mid \{\vec{\lambda}, C\}\big)$$

It should be noted that the likelihoods discussed as part of K_1 and K_2 will be replaced by the log-joint in the software implementation, which slightly changes the structure of the acceptance criteria. In the next part, we discuss how this slight modification still preserves π-invariance.

Part 2: π-Invariance

Prove that the MCMC algorithm you defined in part 1 is π-invariant.

Since detailed balance implies global balance, it is sufficient to show that the DBEs hold to show invariance. We know that if each K_i in the kernel mixture is π-invariant that their mixture is π-invariant. Hence, we will separately prove π-invariance for K_1 and K_2 hold under detailed balance to show that K is invariant.

Conveniently, since K_1 and K_2 share likelihood functions and both have symmetric proposals, the proof of one kernel is equivalent to the proof of the other. In other words, although $\{\vec{\lambda^\star}, C^\star\}$ will have different components changing depending on the kernel, if we prove invariance with respect to $\{\vec{\lambda^\star}, C^\star\}$, this implies invariance of $\{\vec{\lambda}, C^\star\}$ and $\{\vec{\lambda^\star}, C\}$, which can be considered sub-cases where one of two elements is constant.

With this in mind, we let $x^\star = \{\vec{\lambda^\star}, C^\star\}$ be the proposal, $x = \{\vec{\lambda}, C\}$ and K_i be the i-th kernel for $i \in \{1, 2\}$.

As mentioned earlier, we wish to show that detailed balance holds. In other words, we wish to prove that:

$$\pi(x)K_i(x^\star \mid x) = \pi(x^\star)K_i(x \mid x^\star)$$

We will show that the above equality holds by beginning with the left-hand side and simplifying to arrive at the right-hand side.

$$\begin{aligned}
&\pi(x)K_i(x^\star \mid x) = \pi(x) \cdot q_i(x^\star \mid x) \cdot \alpha_i(x^\star, x) \\
&\pi(x)K_i(x^\star \mid x) = \pi(x) \cdot q_i(x^\star \mid x) \cdot \min\{1, r(x^\star, x)\} \\
&\pi(x)K_i(x^\star \mid x) = \pi(x) \cdot q_i(x^\star \mid x) \cdot \min\left\{1, \frac{\gamma(x^\star)}{\gamma(x)}\right\} \\
&\pi(x)K_i(x^\star \mid x) = \pi(x) \cdot q_i(x^\star \mid x) \cdot \min\left\{1, \frac{Z \cdot \pi(x^\star)}{Z \cdot \pi(x)}\right\}, \text{ by definition of } \gamma \\
&\pi(x)K_i(x^\star \mid x) = \pi(x) \cdot q_i(x^\star \mid x) \cdot \min\left\{1, \frac{\pi(x^\star)}{\pi(x)}\right\} \\
&\pi(x)K_i(x^\star \mid x) = q_i(x^\star \mid x) \cdot \pi(x) \cdot \min\left\{1, \frac{\pi(x^\star)}{\pi(x)}\right\}, \text{ by commutativity} \\
&\pi(x)K_i(x^\star \mid x) = q_i(\underset{\downarrow}{x^\star} \mid x) \cdot \min\left\{\pi(x), \pi(x^\star)\right\}, \text{ since } a \cdot \min\{x, y\} = \min\{ax, ay\} \\
&\pi(x)K_i(x^\star \mid x) = \overbrace{q_i(x \mid x^\star)} \cdot \min\left\{\pi(x), \pi(x^\star)\right\}, \text{ by symmetric proposal} \\
&\pi(x)K_i(x^\star \mid x) = q_i(x \mid x^\star) \cdot \left(\frac{\pi(x^\star)}{\pi(x^\star)}\right) \cdot \min\left\{\pi(x), \pi(x^\star)\right\} \\
&\pi(x)K_i(x^\star \mid x) = q_i(x \mid x^\star) \cdot \pi(x^\star)\left(\frac{1}{\pi(x^\star)}\right) \cdot \min\left\{\pi(x), \pi(x^\star)\right\} \\
&\pi(x)K_i(x^\star \mid x) = q_i(x \mid x^\star) \cdot \pi(x^\star) \cdot \min\left\{\frac{\pi(x)}{\pi(x^\star)}, \frac{\pi(x^\star)}{\pi(x^\star)}\right\} \\
&\pi(x)K_i(x^\star \mid x) = \pi(x^\star) \cdot q_i(x \mid x^\star) \cdot \min\left\{\frac{\pi(x)}{\pi(x^\star)}, 1\right\}, \text{ rearranging terms} \\
&\pi(x)K_i(x^\star \mid x) = \pi(x^\star) \cdot q_i(x \mid x^\star) \cdot \min\left\{1, \frac{\pi(x)}{\pi(x^\star)}\right\}, \text{ since } \min\{x, y\} = \min\{y, x\} \\
&\pi(x)K_i(x^\star \mid x) = \pi(x^\star)K_i(x \mid x^\star), \text{ as required} \quad \square
\end{aligned}$$

Therefore, detailed balance holds for $K_i, i \in \{1, 2\}$. This implies that global balance holds for each K_i, which in turn implies that each K_i is π-invariant. Since each K_i in the unified kernel K is π-invariant, the overall kernel is also π-invariant as required.

Part 3: Implementation

Implement the MCMC algorithm you describe mathematically in R.

```
N = nrow(df)
# we build such that dim = 1
mcmc = function(rates, CP, y, n_iterations, debug = FALSE) {
  change_point_trace = rep(-1, n_iterations)
  # initial point
  current_CP = CP
  current_RT = rates
  current = list(rates, CP)
  # iteration station
  for (i in 1:n_iterations) {
    if (debug) {print(unlist(rep("#", times = 10)))}
    if (debug) {print(paste("Iteration:", i))}
    # bernoulli trial
    kernel_choice = ifelse(runif(1) < 0.5, 1, 2)
    if (debug) {print(paste("Chose Kernel", kernel_choice, "..."))}
    # Kernel for Change Point
    if (kernel_choice == 1){
      # Discrete Uniform Proposal
      prop = rdunif(1, 1, N)
      if (debug) {print(paste("Proposed", prop, "..."))}
      # MH ratio
      ratio = (log_joint(current[[1]], prop, y) -
               log_joint(current[[1]], current[[2]], y))
      # catch the indeterminate case
      ratio = ifelse( is.na(ratio), -Inf , ratio)
      if (debug) {print(paste("Ratio", ratio)) }
      # Bernoulli Trial
      if (log(runif(1)) < ratio) {
        # accept
        current[[2]] = prop
      } else {
        # reject (redundant but nice)
        current[[2]] = current[[2]]
      }
    }
    # Kernel for Lambdas
    else {
      # normal at current point
      ell_1 = rnorm(1, mean = current[[1]][1])
      ell_2 = rnorm(1, mean = current[[1]][2])
      # then the proposal is the vector
      prop = c(ell_1, ell_2)
      if (debug) {print(paste("Proposed", unlist(round(prop,2)), "..."))}
      # MH Ratio
      ratio = (log_joint(prop, current[[2]], y) -
               log_joint(current[[1]], current[[2]], y))
      # catch the indeterminate case
      ratio = ifelse( is.na(ratio), -Inf , ratio)

      if (debug) {print(paste("Ratio", ratio)) }
```

```
      if (log(runif(1)) < ratio) {
        # accept
        current[[1]] = prop
      }else {
        # reject (redundant but nice)
        current[[1]] = current[[1]]
      }
      # update trace
     if(debug) {Sys.sleep(1)}
    }
    change_point_trace[i] = current[[2]]
  }
  return(
    list(
      change_point_trace = change_point_trace,
      last_iteration_rates = current[[1]]
    )
  )
}
```

Question 2: MCMC Correctness Testing

We now subject our MCMC implementation to an "exact invariance test" to validate its correctness.

Part 1

We'll start by completing the function below to perform forward simulation on the same Bayesian model as in Q1. The input argument `synthetic_data_size` specifies the size of the dataset to generate.

We recall the model outlined in the previous question. We number each component distribution to provide clear documentation of the forward-sampler.

$$\lambda_1 \sim \exp(1/100) \tag{1}$$
$$\lambda_2 \sim \exp(1/100) \tag{2}$$
$$C \sim \text{unif}\big(\{1, 2, \ldots, 74\}\big) \tag{3}$$
$$Y_d \mid C, \{\lambda_1, \lambda_2\} \sim \text{pois}\big(\mathbb{1}[\,d < C\,]\lambda_1 + \mathbb{1}[\,d \geq C\,]\lambda_2\big) \tag{4}$$

We define the `forward` function given this model below.

```
forward = function(synthetic_data_size) {
  # exponential realization for lambda_1
  lambda_1 = rexp(1, 1/100) # (1)
  # exponential realization for lambda_2
  lambda_2 = rexp(1, 1/100) # (2)
  # discrete uniform for C
  change_point = rdunif(1, 1, 74) # (3)
  # define vector for data
  data = sapply(1:synthetic_data_size,
    # simulate from index
    function(d){
      # likelihood for Y_d
```

```
      rpois(1, ifelse(d < change_point, lambda_1, lambda_2)) # (4)
    })
  # return a list of all components
  return(list(
    rates = c(lambda_1, lambda_2),
    change_point = change_point,
    data = data
  ))
}
# forward(100)
```

Now, we import the provided `forward_posterior` function:

```
forward_posterior = function(synthetic_data_size, n_mcmc_iters) {
  # sample from the forward simulator
  initial = forward(synthetic_data_size)
  # in the case we are using mcmc
  if (n_mcmc_iters > 0) {
    # run the mcmc algorithm on this forward-sampled initial point
    samples = mcmc(initial$rates, initial$change_point,
                   initial$data, n_mcmc_iters)
    # return pre-change point rate after mcmc
    return( samples$last_iteration_rates[[1]] )
  }
  # otherwise simply return simulated lambda
  else {
    return(initial$rates[[1]])
  }
}
```

Finally, we run the Kolmogorov-Smirnov test for the chosen iteration counts:

```
set.seed(1928)
# NOTE: we use synthetic datasets with only 5 observations to speed things up
forward_only = replicate(1000, forward_posterior(5, 0))
with_mcmc = replicate(1000,    forward_posterior(5, 200))
# run test
ks.test(forward_only, with_mcmc)
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  forward_only and with_mcmc
## D = 0.024, p-value = 0.9356
## alternative hypothesis: two-sided
```

We see that the observed p-value of the Kolmogorov-Smirnov test is approximately 0.9356. Under the assumption H_0, the probability of observing discrepancies as large or larger than those observed between the empirical CDFs of `forward_only` and `with_mcmc` is approximately 93.56%. This high p-value is a good sign that the code is functioning as intended.

Part 3

If you identified a bug using this method, show the part of the code that had a bug (before and after fixing), as well as the p-value (before and after). If no bugs where found, temporarily create one and report the same thing. Don't forget to fix all bugs before moving to the next question.

While I did in fact identify a bug using this method, it didn't have a actually prevented the algorithm from running correctly.

Specifically, the reported error was "`Error in if (log(runif(1)) < ratio) { : missing value where TRUE/FALSE needed`" which was occurring when the starting C value was beyond the index of any of the observations. I added a line returning `-Inf` in the case that both elements of the ratio were not well-defined.

To show the utility of the Kolmogorov-Smirnov, I manually injected a bug into the code. Specifically, I set `log = FALSE` for the likelihood value in the `log_joint` function. In this case, the test yielded a p-value of `0.03328`.

Question 3 : Data Analysis

Perform 10k iteration of MCMC on the text message data.

First, we select candidate starting points $\vec{\lambda}^{(0)}$ and $C^{(0)}$. I picked $C^{(0)} = 37$ (halfway point) and $\vec{\lambda^{(0)}} = \{1, 1\}$

```
M = 10000
rates_0 = c(1, 1)
C_0 = 37
```

```
set.seed(007) # The name's Chain... Markov Chain.
samples = mcmc(rates_0, C_0, y = sms_data, n_iterations = M)
```

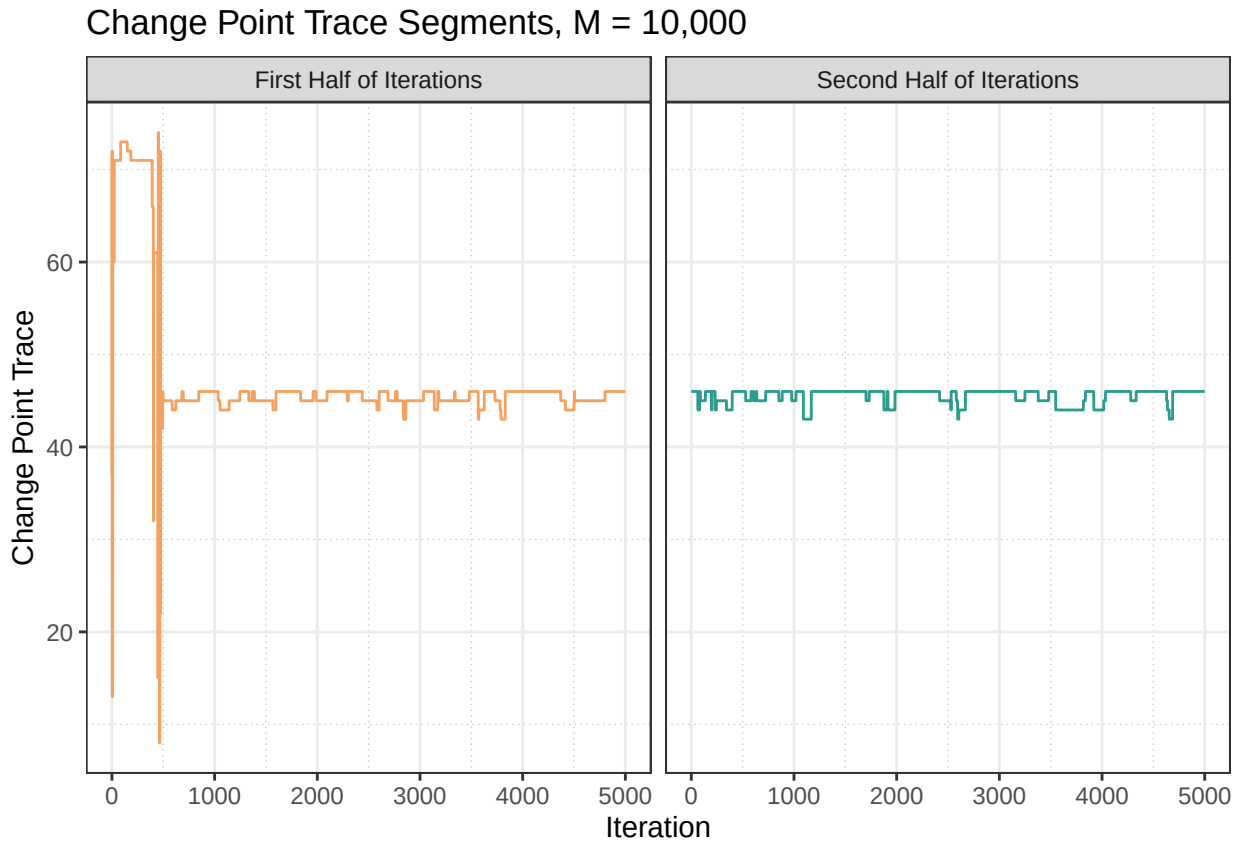
Part 1

Create two trace plots for the change point parameter, one for the full trace, one for the subset of samples in the second half. Comment on the mixing behaviour.

```
# first half
dDF1 <- data.frame(Index = 1:5000,
                   ChangePointTrace = samples$change_point_trace[1:5000],
                   Segment = 'First Half of Iterations')
# second half
dDF2 <- data.frame(Index = 0:5000,
                   ChangePointTrace = samples$change_point_trace[5000:10000],
                   Segment = 'Second Half of Iterations')

# prep for plotting
dDF <- bind_rows(dDF1, dDF2)
# create plot
p <-
  ggplot(dDF, aes(x = Index, y = ChangePointTrace, color = Segment)) +
  geom_line() +
  facet_wrap( ~ Segment, ncol = 2) +
  theme_bw() +
```

```
  scale_colour_manual(values = c(
    'First Half of Iterations' = "#f4a261",
    'Second Half of Iterations' = "#2a9d8f"
  )) +
  labs(x = "Iteration", y = "Change Point Trace",
       title = "Change Point Trace Segments, M = 10,000") +
  theme(
    panel.grid.minor = element_line(color = "grey80", linetype = 'dotted'),
    legend.position = "none"
  )
# show plot
print(p)
```



Comments: The chain seems to be mixing well, and mixing quickly at that! This is encouraging behaviour. We see a lot of variation and movement for approximately the first 500 iterations, but as $M \uparrow$, the Change Point traces seem to converge rather well and surround some value in the mid-40s. Both the combination of the initial variation tailing off and the MCMC surrounding some fixed value are good signs that the chain is mixing well.

Part 2:

We report the effective sample size for the full set of samples and each of the subsets.

```
quiet_load("mcmcse")

kable(t(data.frame(ESS = round(c(
  ess(dDF$ChangePointTrace),
  ess(dDF1$ChangePointTrace),
  ess(dDF2$ChangePointTrace)
), 2))),
col.names = c("Full Set", "First Half", "Second Half"),
caption = "Effective Sample Sizes for MCMC"
)
```

Table 1: Effective Sample Sizes for MCMC

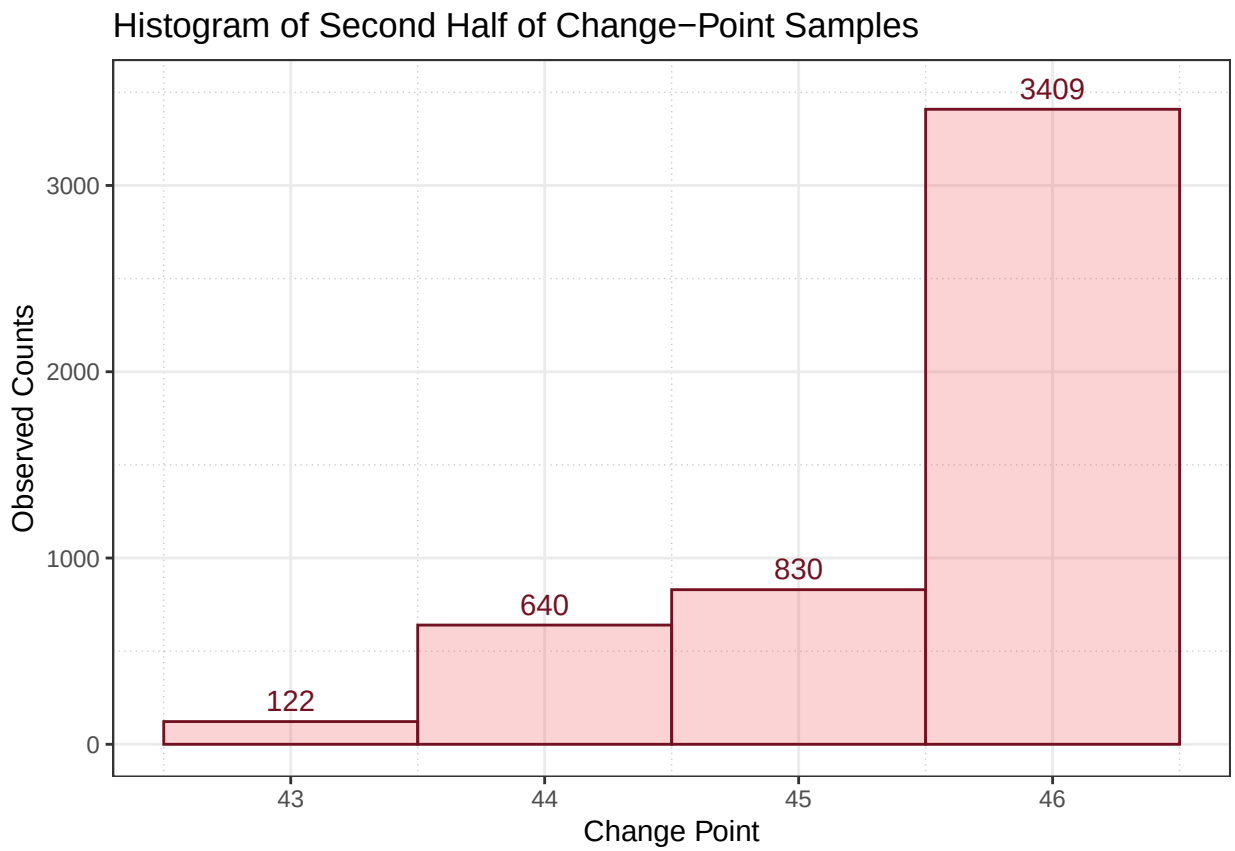
	Full Set	First Half	Second Half
ESS	42.29	25.44	43.39

As the above table shows, the effective sample size is slightly larger than the second half of the samples than the overall set. I decided to include the ESS of the first half as well for comparison, and we can see the effective sample size is much larger for the second half of the samples as well, which helps reinforce our previous conclusion of a well-mixing algorithm.

Part 3

Produce a histogram from the second half of samples.

```
p2 <- ggplot(dDF2, aes(x = ChangePointTrace)) +
  geom_histogram(
    binwidth = 1,
    fill = "#F15156",
    linewidth = 0.5,
    color = "#721121",
    alpha = 0.25
  ) +
  geom_text(
    stat = 'count',
    aes(label = after_stat(count), y = after_stat(count)),
    vjust = -0.5,
    color = "#721121"
  ) +
  ylim(0, 3500) +
  labs(title = "Histogram of Second Half of Change-Point Samples",
       y = "Observed Counts",   x = "Change Point") +
  theme_bw() +
   theme(
    panel.grid.minor = element_line(color = "grey80", linetype = 'dotted'))
print(p2)
```



Briefly comment on the results in light of the note below.

Note: this dataset comes with the additional information that *"the 45th day corresponds to Christmas, and I moved away to Toronto the next month, leaving a girlfriend behind"* (Davidson-Pilon, 2013).

With this additional information, the large number of samples of 46 begins to make sense. From this quote, it is likely that the author's life changed significantly in the time following Christmas, implying that the 46th day likely corresponds to the change point.