Excercise 2, Spring 2022

TMA4300 - Computer Intensive Statistical Methods

Sivert Laukli, Thomas Torkildsen

3 3 2022

Introduction

In this report we will apply the methods of Markov chain Monte-Carlo simulation (MCMC) and integrated nested Laplace approximation (INLA) on the Tokyo rainfall dataset containing daily rainfall data from 1951-1989. The variable t will represent the index of the day of the year, and n_t will represent the number of such days, where $n_t = 39$ for all $t \in \{1, ...59, 61, ..., 366\}$ and $n_t = 10$ for $t = 60$. For further mentions of t, we let $t = 1, ..., T$, where $T = 366$.

We will in the first problem sections derive some theoretical results based on a hierarchical Bayes model approach which will be used to a smaller or larger degree in the latter sections. Then we will implement a MCMC sampler updating every element individually using Metropolis-Hastings steps, and then implement a MCMC sampler updating blocks of elements also using Metropolis-Hastings steps. In the last section we will obtain the same inference using INLA. We will continuously throughout the report compare the performances and results and lastly conclude which model performs the best.

In our programmed functions, we will apply some log-scaled calculations as multiplication of log-scaled values are simpler than multiplications in R. In these calculation we will take the exponential functions of the values at the end finding the final results. Therefore we will also give the logarithm of the results in our analytical derived expressions.

Problem 1

In this problem the response variable will be wether or not the amount of rain fallen in Tokyo on any given day will exceed 1 mm over the time period of the data, where

$$y_t|\tau_t = y_t|\pi(\tau_t) \sim \text{Bin}(n_t, \pi(\tau_t))$$

is conditionally independent for all $t = 1, ..., T$, and

$$\pi(\tau_t) = \frac{\exp(\tau_t)}{1 + \exp(\tau_t)} = \frac{1}{1 + \exp(-\tau_t)}, \log[\pi(\tau_t)] = -\log[1 + \exp\{-\tau_t\}].$$

a)

We begin with loading the Tokyo rainfall data set.

```r
load(file = "rain.rda")
```

We now explore the data plotting the probability of rain above 1 mm as a function of time in days. As we have 10 samples from day 60 and 39 from the rest, we plot the probability of rain rather than the number of rainy days. Thus we first define some new variables in our data. To obtain the original data `rain` unchanged, we create a copy `r`.

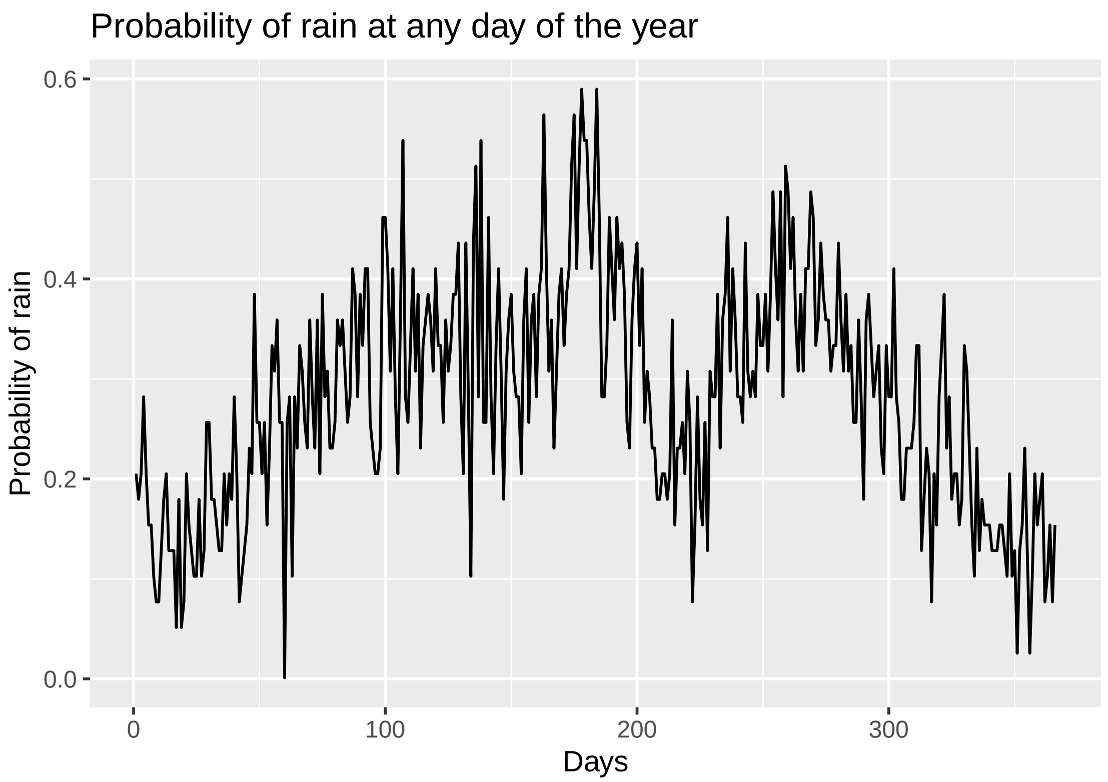
```
r = rain #copying rain into a new variable
r$prob_rain = r$n.rain/r$n.years #calculating the estimated probability of rain
```

As some of the values from `r$prob_rain` are zero and we are going to perform logarithmic transforms on this data later, we set the zero values to a number close to zero that is suitable for the log-transform for later.

```
init_fix <- function(p){
  p = ifelse(p == 0, 0.001, p)
  return(p)
}
r$prob_rain = init_fix(r$prob_rain)
```

We now plot the probability.

```
library(ggplot2)

(plot1 = ggplot(r, aes(day, prob_rain)) + geom_line() +
    labs(title = "Probability of rain at any day of the year",
         caption = "Probability of rain at any day of the year, given by the Tokyo rainfall dataset.",
         tag = "Figure 1") + xlab("Days") + ylab("Probability of rain") +
    theme(plot.tag.position = "bottomleft"))
```



Probability of rain at any day of the year, given by the Tokyo rainfall dataset.

Figure 1

We see in Figure 1 that it on average is less probability of rain in Tokyo at the start and at the end of the year. Furthermore, the probability peaks around day 180 which is at the end of June. The rainfall probability fluctuates around approximately 0.35 from the beginning of April until the middle of June, then it rises to the peak of approximately 0.58 at the end of June, before it drops to approximately approximately 0.15 at the end of July. Then it rises to around 0.4 around mid-September. Towards the end of the year, the rainfall probability again falls to approwimately 0.15.

b)

We now want the likelihood of y_t depending on the parameters $\pi(\tau_t)$ for $t = 1, ..., 366$. Defining the probability distribution function of $y_t|\pi(\tau_t)$ as

$$f(y_t|\pi(\tau_t)) = \binom{n_t}{y_t}\pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{(n_t-y_t)},$$
$$\log[f(y_t|\pi(\tau_t))] = \log\left[\binom{n_t}{y_t}\right] + y_t \cdot \log[\pi(\tau_t)] + (n_t - y_t)\cdot\log[1-\pi(\tau_t)],$$

we obtain the likelihood by writing the joint distribution of $y_t|\pi(\tau_t)$ explicitly as follows.

$$
\begin{aligned}
L(\mathbf{y}|\pi(\tau)) &= \prod_{t=1}^{T} f(y_t|\pi(\tau_t)) \\
&= \prod_{t=1}^{T} \binom{n_t}{y_t} \pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{(n_t-y_t)}, \\
\log[L(\mathbf{y}|\pi(\tau))] = l(\mathbf{y}|\pi(\tau)) &= \sum_{t=1}^{T} \left[\log\left[\binom{n_t}{y_t}\right] + y_t \cdot \log[\pi(\tau_t)] + (n_t - y_t)\cdot \log[1-\pi(\tau_t)]\right].
\end{aligned}
$$

A hierarchial Bayesian model will now be applied to the dataset. A random walk of order 1 (RW(1)) will be used to model the trend on a logit scale, i. e. in terms of τ_t, where

$$\tau_t \sim \tau_{t-1} + u_t,$$

$$u_t \sim_{iid} N(0, \sigma_u^2).$$

The parameter σ_u^2 will have an inverse gamma prior (IG(α,β)) with shape parameter α and scale parameter β given as

$$
\begin{aligned}
p(\sigma_u^2) &= \frac{\beta^\alpha}{\Gamma(\alpha)}(1/\sigma_u^2)^{\alpha+1}\exp\left\{-\frac{\beta}{\sigma_u^2}\right\}, \\
\log[p(\sigma_u^2)] &= \alpha\log[\beta] - \log[\Gamma(\alpha)] + (\alpha+1)\cdot(-\log[\sigma_u^2]) - \frac{\beta}{\sigma_u^2}.
\end{aligned}
$$

Let now $\mathbf{y} = (y_1, ..., y_T)^T$, $\tau = (\tau_1, ..., \tau_T)^T$ and $\pi = (\pi(\tau_1), ..., \pi(\tau_T))$.

c)

We now want the conditional distribution $p(\sigma_u^2|\mathbf{y}, \tau)$. This is the product of $L(\mathbf{y}|\pi(\tau))$, $p(\sigma_u^2)$ and the distribution of $\tau|\sigma_u^2$. The latter can be defined as

$$
\begin{aligned}
p(\tau|\sigma_u^2) &= \prod_{t=2}^{T} \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left\{-\frac{1}{2\sigma_u^2}(\tau_t - \tau_{t-1})\right\} \\
&= \left(\frac{1}{2\pi\sigma_u^2}\right)^{\frac{T-1}{2}} \exp\left\{-\frac{1}{2\sigma_u^2}\sum_{t=2}^{T}(\tau_t - \tau_{t-1})\right\}, \\
\log[p(\tau|\sigma_u^2)] &= \left(\frac{T-1}{2}\right)\cdot(-\log[2\pi\sigma_u^2]) - \frac{1}{2\sigma_u^2}\sum_{t=2}^{T}(\tau_t - \tau_{t-1})
\end{aligned}
$$

since $\tau_t \sim \tau_{t-1} + u_t$ and $u_t \sim_{iid} N(0, \sigma_u^2)$.

Thus the conditional distribution of $\sigma_u^2|\mathbf{y},\tau$ can be written as

$$\begin{aligned}
p(\sigma_u^2|\mathbf{y},\tau) &= L(\mathbf{y}|\pi(\tau))\cdot p(\sigma_u^2)\cdot p(\tau|\sigma_u^2)\\
&= \prod_{t=1}^{T}\binom{n_t}{y_t}\pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{n_t-y_t}\cdot\frac{\beta^\alpha}{\Gamma(\alpha)}(1/\sigma_u^2)^{\alpha+1}\exp\left\{-\frac{\beta}{\sigma_u^2}\right\}\cdot\left(\frac{1}{2\pi\sigma_u^2}\right)^{\frac{T-1}{2}}\exp\left\{-\frac{1}{2\sigma_u^2}\sum_{t=2}^{T}(\tau_t-\tau_{t-1})\right\}\\
&\propto \left(\frac{1}{\sigma_u^2}\right)^{\alpha+\frac{T-1}{2}+1}\cdot\exp\left\{-\frac{1}{\sigma_u^2}\left[\frac{1}{2}\sum_{t=2}^{T}(\tau_t-\tau_{t-1})^2+\beta\right]\right\}\\
&= \left(\frac{1}{\sigma_u^2}\right)^{\alpha+\frac{T-1}{2}+1}\cdot\exp\left\{-\frac{1}{\sigma_u^2}\left[\frac{1}{2}\cdot\tau\mathbf{Q}\tau^{\mathbf{T}}+\beta\right]\right\},\\
\log[p(\sigma_u^2|\mathbf{y},\tau)] &\propto \left(\alpha+\frac{T-1}{2}+1\right)\cdot(-\log[\sigma_u^2])-\frac{1}{\sigma_u^2}\left[\frac{1}{2}\cdot\tau\mathbf{Q}\tau^{\mathbf{T}}+\beta\right].
\end{aligned}$$

As an inverse gamma distribution with shape a and scale b has the PDF

$$\mathrm{IG}(a,b)=\frac{b}{\Gamma(a)}\left(\frac{1}{x}\right)^{a+1}\exp\left\{-\frac{b}{x}\right\},$$

we see that the our $p(\sigma_u^2|\mathbf{y},\tau)$ represents the core of a inverse gamma function with shape $a=\alpha+\frac{T-1}{2}$ and scale $b=\frac{1}{2}\sum_{t=2}^{T}(\tau_t-\tau_{t-1})^2+\beta$, i. e.

$$\sigma_u^2|\mathbf{y},\tau\sim\mathrm{IG}\left(\alpha+\frac{T-1}{2},\frac{1}{2}\cdot\tau\mathbf{Q}\tau^{\mathbf{T}}+\beta\right).$$

d)

We will now consider the consitional prior distribution $Q({\tau'}_i|\tau_{-i},\sigma_u^2,\mathbf{y})=p({\tau'}_i|\tau_{-i},\sigma_u^2)$, where ${\tau'}_i$ is the proposed value for τ_i, $i\subseteq\{1,...,T\}$ is the set of indices and $\tau_{-i}=\tau_{\{1,...,T\}\setminus i}$ is a subset of τ including all other indices of τ but index i. We shall now use this and the earlier derived expressions to show that the acceptance probability is given by the ratio of the likelihoods,

$$\alpha({\tau'}_i|\tau_{-i},\sigma_u^2,\mathbf{y})=\min\{1,\frac{p(\mathbf{y}_i|{\tau'}_i)}{p(\mathbf{y}_i|\tau_i)}\}=\min\{1,a\},$$

where a is defined as the ratio above, and by definition is given as

$$a=\frac{p(\tau',\sigma_u^2|\mathbf{y})\cdot Q(\tau_i|\tau_{-i},\sigma_u^2,\mathbf{y})}{p(\tau,\sigma_u^2|\mathbf{y})\cdot Q({\tau'}_i|\tau_{-i},\sigma_u^2,\mathbf{y})}=\frac{p(\tau',\sigma_u^2|\mathbf{y})\cdot p(\tau_i|\tau_{-i},\sigma_u^2)}{p(\tau,\sigma_u^2|\mathbf{y})\cdot p({\tau'}_i|\tau_{-i},\sigma_u^2)}$$

$p(\tau',\sigma_u^2|\mathbf{y})$ can then be rewritten as

$$p(\tau',\sigma_u^2|\mathbf{y})=\frac{p(\tau',\mathbf{y},\sigma_u^2)}{p(\mathbf{y})}\propto p(\mathbf{y}|\tau',\sigma_u^2)\cdot p(\tau'|\sigma_u^2)\cdot p(\sigma_u^2).$$

Let now $\tau'=({\tau'}_i,\tau_{-i})$. Then

$$p(\mathbf{y}|\tau',\sigma_u^2)\cdot p(\tau'|\sigma_u^2)\cdot p(\sigma_u^2)=p(\mathbf{y}|\tau',\sigma_u^2)\cdot p({\tau'}_i|\tau_{-i},\sigma_u^2)\cdot p(\tau_{-i}|\sigma_u^2)\cdot p(\sigma_u^2).$$

Likewise for $p(\tau,\sigma_u^2|\mathbf{y})$,

$$p(\tau,\sigma_u^2|\mathbf{y})=\frac{p(\tau,\mathbf{y},\sigma_u^2)}{p(\mathbf{y})}\propto p(\mathbf{y}|\tau,\sigma_u^2)\cdot p(\tau_i|\tau_{-i},\sigma_u^2)\cdot p(\tau_{-i}|\sigma_u^2)\cdot p(\sigma_u^2).$$

Thus a becomes

$$a=\frac{\frac{p(\mathbf{y}|\tau',\sigma_u^2)\cdot p({\tau'}_i|\tau_{-i},\sigma_u^2)\cdot p(\tau_{-i}|\sigma_u^2)\cdot p(\sigma_u^2)\cdot p(\tau_i|\tau_{-i},\sigma_u^2)}{p(\mathbf{y})}}{\frac{p(\mathbf{y}|\tau,\sigma_u^2)\cdot p(\tau_i|\tau_{-i},\sigma_u^2)\cdot p(\tau_{-i}|\sigma_u^2)\cdot p(\sigma_u^2)\cdot p({\tau'}_i|\tau_{-i},\sigma_u^2)}{p(\mathbf{y})}}=\frac{p(\mathbf{y}|\tau',\sigma_u^2)}{p(\mathbf{y}|\tau,\sigma_u^2)}.$$

In our case, $\mathbf{y}_i$ and $\mathbf{y}_{-i}$, where the subscripts have the same functions as for τ, are conditionally independent. Then

$$a = \frac{p(\mathbf{y}_i|\tau', \sigma_u^2) \cdot p(\mathbf{y}_{-i}|\tau', \sigma_u^2)}{p(\mathbf{y}_i|\tau, \sigma_u^2) \cdot p(\mathbf{y}_{-i}|\tau, \sigma_u^2)}.$$

As well, in our case, $\mathbf{y}_i$ is independent of τ_{-i}, and $\mathbf{y}_{-i}$ is independent of τ_i and ${\tau'}_i$. This gives

$$a = \frac{p(\mathbf{y}_i|{\tau'}_i, \sigma_u^2) \cdot p(\mathbf{y}_{-i}|\tau_{-i}, \sigma_u^2)}{p(\mathbf{y}_i|\tau_i, \sigma_u^2) \cdot p(\mathbf{y}_{-i}|\tau_{-i}, \sigma_u^2)} = \frac{p(\mathbf{y}_i|{\tau'}_i, \sigma_u^2)}{p(\mathbf{y_i}|\tau_i, \sigma_u^2)}.$$

As both the expressions in the ratio above does not contain σ_u^2, the ratio a becomes

$$a = \frac{p(\mathbf{y}_i|{\tau'}_i)}{p(\mathbf{y_i}|\tau_i)},$$

such that the acceptance probability becomes

$$\alpha({\tau'}_i|\tau_{-i}, \sigma_u^2, \mathbf{y}) = \min\{1, a\} = \min\{1, \frac{p(\mathbf{y}_i|{\tau'}_i)}{p(\mathbf{y}_i|\tau_i)}\},$$

which is what we wanted to show.

e)

After having defined the distributions above, we now use the fact that the precision matrix is defined as

$$\text{Prec}(\tau|\sigma_u^2) = \frac{1}{\sigma_u^2}\mathbf{Q},$$

where

$$\mathbf{Q} = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix},$$

which can be derived from $p(\tau|\sigma_u^2)$. Furthermore, we are given the properties

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_B \end{bmatrix} \sim MVN\left(\begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix}, \begin{bmatrix} \mathbf{Q}_{AA} & \mathbf{Q}_{AB} \\ \mathbf{Q}_{BA} & \mathbf{Q}_{BB} \end{bmatrix}^{-1} \right),$$

where conditional mean and precision are given as, when assuming $\mu \mathbf{1}_A = \mu_A$ and $\mu \mathbf{1}_B = \mu_B$

$$\mu_{A|B} = \mu_A - \mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}(\mathbf{x}_B - \mu_B) = \mu \mathbf{1}_A - \mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}\mathbf{x}_B + \mu \mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}\mathbf{1}_B = -\mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}\mathbf{x}_B,$$

and

$$\mathbf{Q}_{A|B} = \mathbf{Q}_{AA}.$$

Assuming a similar expected value for the multivariate normal distribution, the conditional proposal prior distribution $p(\tau_i|\tau_{-i}, \sigma_u^2)$ becomes

$$\tau_{\mathbf{i}}|\tau_{-i}, \sigma_u^2 \sim MVN\left(-\mathbf{Q}_{i,i}^{-1} \cdot \mathbf{Q}_{i,-i} \cdot \tau_{-i}, \sigma_u^2 \cdot \mathbf{Q}_{i,i}^{-1}\right),$$

We will now use this to create a MCMC sampler sampling from a target distribution $p(\pi, \sigma_u^2|\mathbf{y})$. We will use the Metropolis-Hastings algorithm by sampling values of τ from the conditional prior $p(\tau_t|\tau_{-t}, \sigma_u^2)$, and Gibbs steps to sample σ_u^2 from $p(\sigma_u^2|\mathbf{y}, \tau)$, which we have shown to be inverse gamma distributed with shape parameter $\alpha + (T-1)/2$ and rate $\beta + \frac{1}{2}\tau\mathbf{Q}\tau^T$, based on the samples of τ.

For the Metropolis-Hastings algorithm, we use a normal distributed proposal with mean and variance as described above. We sample from a proposal value from this distribution with mean equal to the value of the previous step, and calculate the acceptance probability according to what we found under section d), i. e. the minimum of 1 and the ratio of the previous likelihood and the proposed likelihood. If the proposal is accepted, we update the value to the proposal and if it is rejected we update it to the previous value. For the Gibbs update of σ_u^2, we update it after the whole conditional vector τ is updated and calculate a new approximation conditioning on the previous τ before every iteration.

Before we create the MCMC sampling function, we define some variables and functions that is helpful throughout the sampling. We first create the variable of τ given its definition and our dataset, as well as a function returning the logarithm of our obtained likelihoodfunction, $l(\mathbf{y}|\pi(\tau))$. As well we create a function returning the matrix $\mathbf{Q}$ defined as above.

```
r$tau = log(r$prob_rain) - log(1-r$prob_rain) # tau from the dataset

#log likelihood function for a joint binomial
log_likelihood <- function(n,y,p) {
  lbin = sum(lchoose(n,y) + y*log(p) + (n-y)*log(1-p))
  return(lbin)
}

#defining Q, without sigma
Q_def <- function(n=366){
  Q <- diag(2, n)
  Q[1,1] <- Q[n,n] <- 1
  Q[abs(row(Q)-col(Q)) == 1] = -1
  return(Q)
}
```

We now have what is needed to implement the MCMC sampler, which is done below. We use the initial value of $\pi(\tau)$ as $\frac{y}{n}$ from our dataset, where y is the number of days where the amount of rain exceeds 1 mm and n is the total amount of such days measured. For the Gibbs step, we are given $\alpha = 2$ and $\beta = 0.05$. We save the MCMC samples in the variable `x`. To calculate the running time of the sampler, we use the `proc.time()[3]` function and taking the difference of its value before and after calling the function.

Since matrix multiplications are computationally expensive and since the precision matrix is sparse, the matrix multiplications are simplified as follows.

$$\mathbf{Q}_{t,t} = \begin{cases} 1 & t = \{1, T\} \\ 2 & t = \{2, ..., T-1\} \end{cases}$$

$$-\mathbf{Q}_{t,t}^{-1}\mathbf{Q}_{t,-t}\tau_{-t} = \begin{cases} \tau_2 & t = 1 \\ (\tau_{t-1} + \tau_{t+2})/2 & t = \{2, ..., T-1\} \\ \tau_{T-1} & t = T \end{cases}.$$

Using the results above, we implement the MCMC sampler as follows.

```
library(invgamma)
mcmc_RW = function(ntimes = 1000)
{

  te = 366            #number of days
  Q = Q_def(te)       #366x366 precition matrix, excluding sigma_u
  # initial values
  init_pi = r$prob_rain #init probability of rain
  init_n = r$n.years #number of years for each day
  init_y = r$n.rain #number of days with rain more than 1mm
  init_tau = r$tau #initial tau value
  tau = matrix(nrow = ntimes, ncol = te) #creating a n x T matrix

  tau[1,] = init_tau #insering the initial tau into the matrix
  sigmas = vector(length = ntimes-1)

  alpha = 2 #given alpha value
  beta = 0.05 #given beta value
  count = 0 #counting the number of accepted values


  for(i in 2:ntimes) #starting the for loop
  {

    sigmas[i-1] = rinvgamma(1, shape = alpha + (te-1)/2, rate = 1/2*sum(diff(tau[i-1,])^2)+beta) #findi

    #loop for estimating each tau
    for(j in 1:te){
      if(j == 1) {
        mu = tau[i-1,2]
        var = sigmas[i-1]
      }
      else if(j == te) {
        mu = tau[i-1,te-1]
        var = sigmas[i-1]
      }
      else {
        mu = -1/2*(Q[j,j-1]*tau[i-1,j-1] + Q[j,j+1]*tau[i-1,j+1])
        var = sigmas[i-1]/2
      }
      temp = rnorm(1, mean = mu, sd = sqrt(var))
      temp_pi = 1/(1+exp(-temp))
      ratio = log_likelihood(init_n[j], init_y[j], temp_pi) - log_likelihood(init_n[j], init_y[j], 1/(1
      log_alpha = min(0, ratio)
      if(log(runif(1))< log_alpha){
        tau[i,j] = temp
        count = count + 1 #counting total number of times accepted
      }
      else
        tau[i,j] = tau[i-1,j]
    }
  }
  acceptrate = count/(ntimes*te) #general acceptance rate
  ret = list(tau = tau, rate = acceptrate, sigmas = sigmas, pi = 1/(1+exp(-tau)))
  return(ret)
}


pti <- proc.time()[3]
```

```r
x = mcmc_RW(n=50000)
ptf <- proc.time()[3]
pt <- ptf - pti

pidf = as.data.frame(x$pi)
```

Running the algorithm with $n = 50000$ iterations yelds the acceptance rate

```r
print.data.frame(data.frame("Acceptance.rate" = x$rate))
```

```
##   Acceptance.rate
## 1       0.7565116
```

which is something that we could expect.

The running time of our sampler is

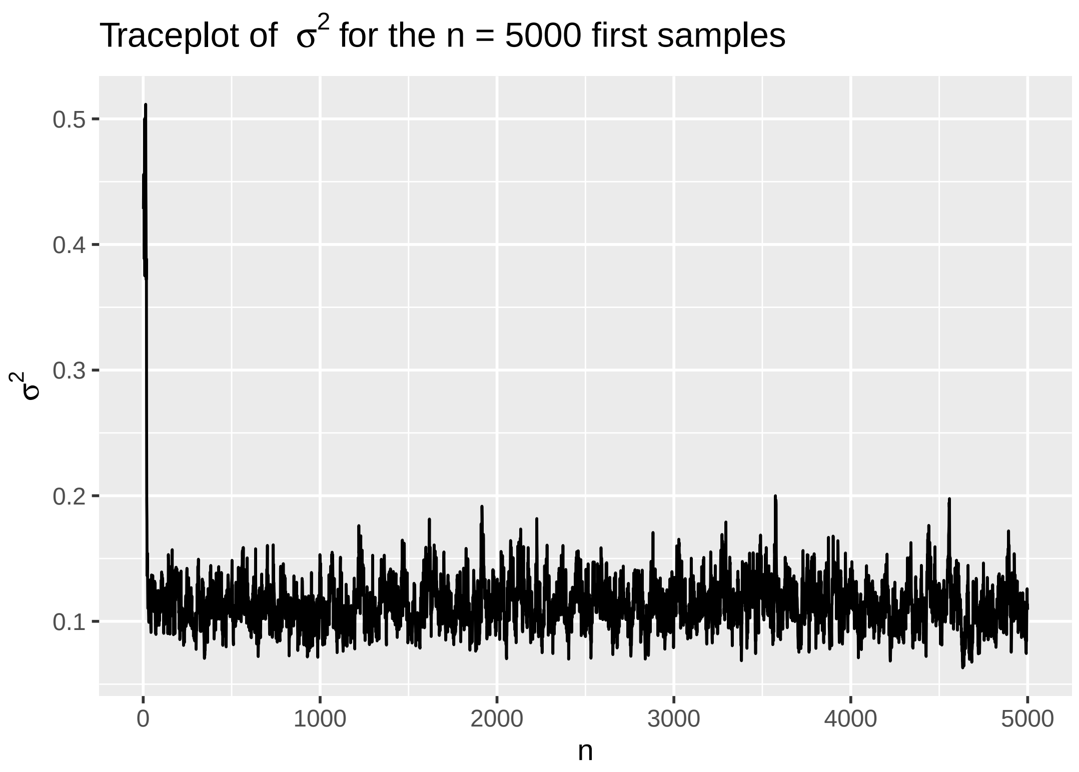
```r
pt
```

```
## elapsed
##  528.39
```

which is rather unpractical for our purpose. There is most probably several ways of doing this more efficient.

We now create plots and get inference about the magnitudes we are estimating. First we consider σ_u^2.

```r
sigmadftp <- data.frame("Sigma" = x$sigmas[1:5000], "n" = 1:5000)
(traceplot_sigma <- ggplot(sigmadftp, aes(n, Sigma)) + geom_line() +
    labs(title = bquote("Traceplot of " ~ sigma ^ 2 ~ "for the n = 5000 first samples"),
         caption = bquote("The figure shows a traceplot of " ~ sigma ^ 2 ~ "for the n = 5000 first sampl
         tag = "Figure 2.1") + ylab(bquote(sigma ^ 2)) +
    theme(plot.tag.position = "bottomleft", plot.caption.position = "panel"))
```



The figure shows a traceplot of σ^2 for the n = 5000 first samples.

Figure 2.1

```r
sigmadf = data.frame("Sigma" = x$sigmas, "n" = 1:49999)
(histogram_sigma = ggplot(sigmadf, aes(Sigma)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Histogram of" ~ sigma ^2),
       caption = bquote("The figure shows a histogram of "  ~ sigma ^ 2),
       tag = "Figure 2.2") + ylab("n") + xlab(bquote(sigma^2)) +
  theme(plot.tag.position = "bottomleft"))
```

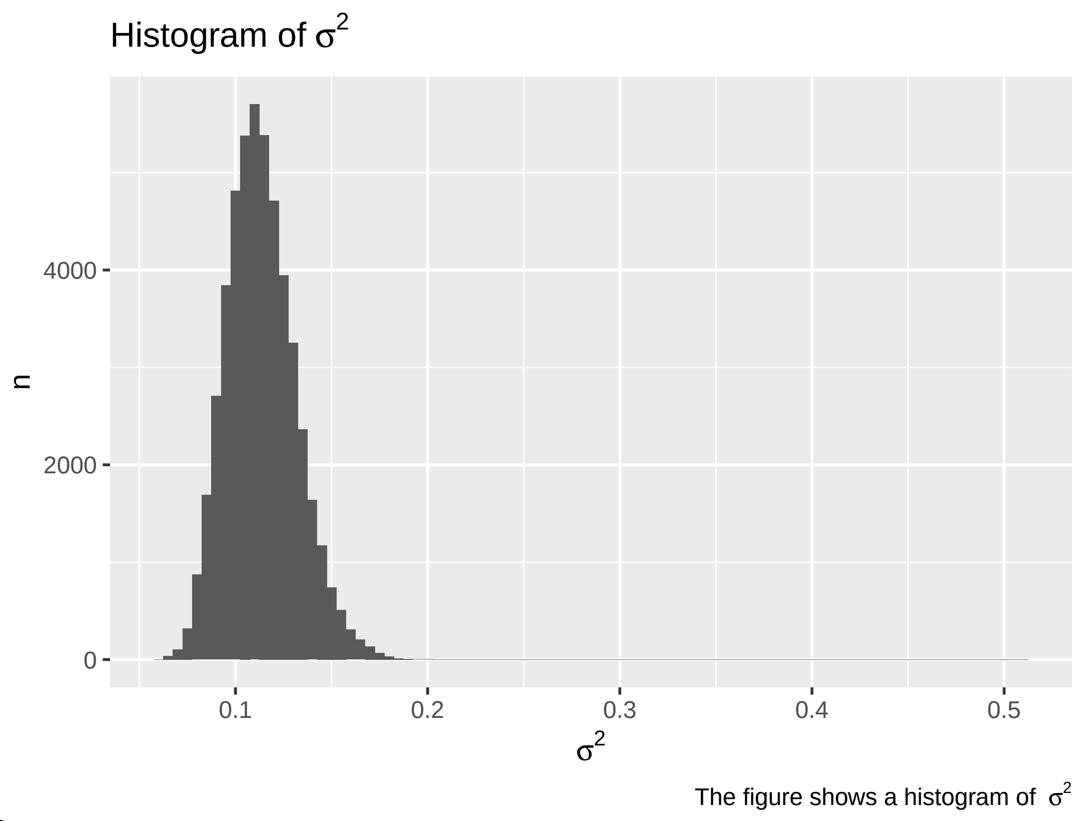


Figure 2.2

```r
bacf = acf(sigmadf$Sigma, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(sigma_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ sigma ^2),
       caption = bquote("The figure shows the autocorrelation function of "  ~ sigma ^ 2),
       tag = "Figure 2.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```

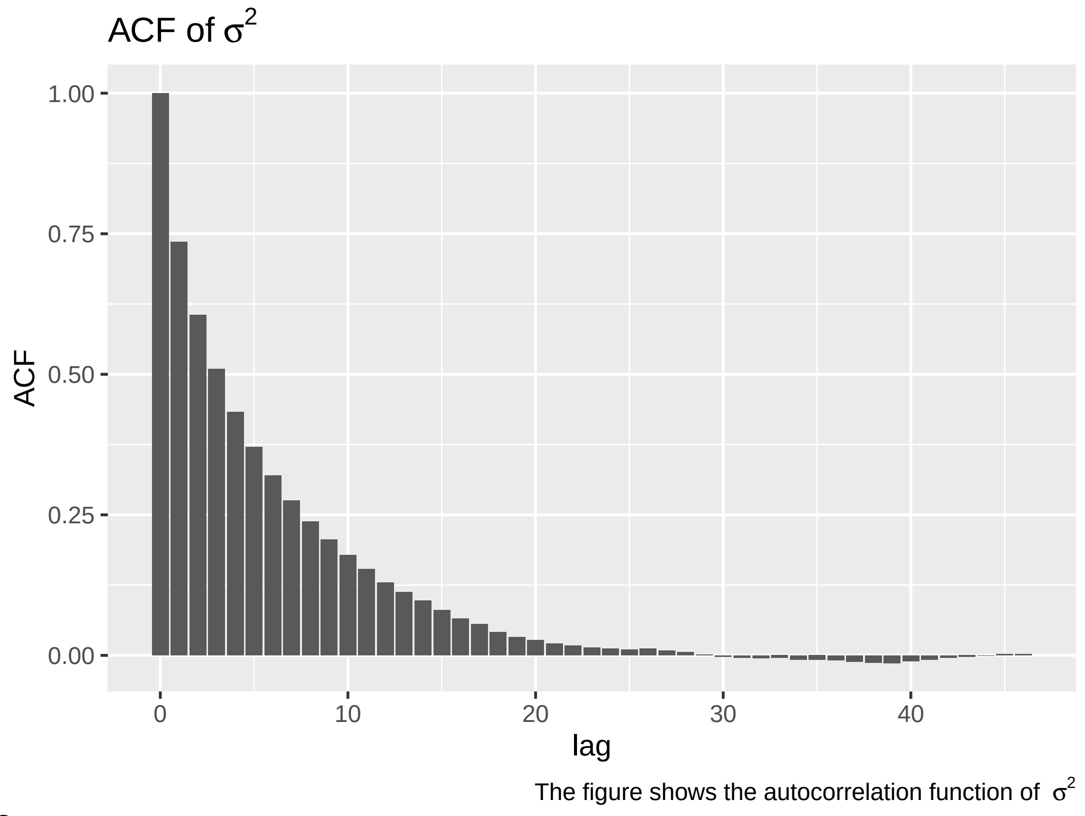


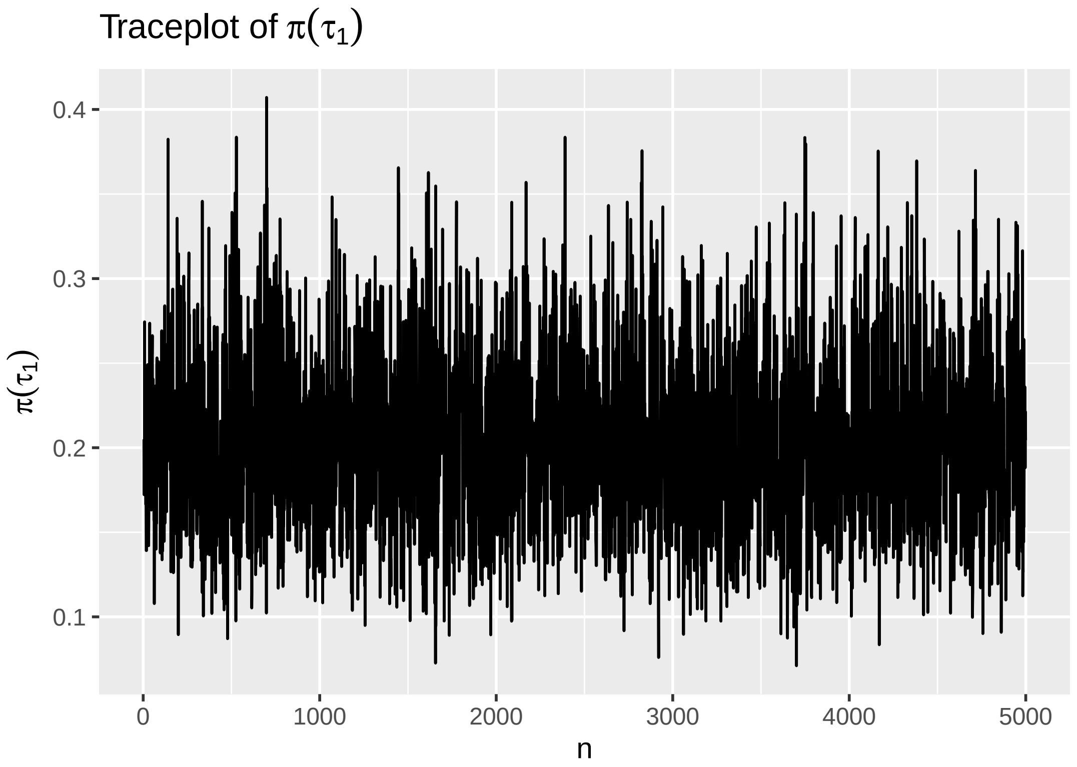
Figure 2.3

As σ_u^2 is sampled from an inverse gamma distribution with continuously updated contitionals, we would expect its MCMC approximation using Gibbs to converge together with the approximated values of τ. From the traceplot visualized in Figure 2.1 we see that the estimator for σ_u^2 has a relatively short burn-in period. Its values are concentrated mainly between 0.01 to 0.2 seen in the histogram in Figure 2.2, which fits well with the assumption that 95% of the prior mass should lie within 0.01 and 0.25. The autocorrelation function in Figure 2.3 shows a somewhat exponential decay, which gives the impression of less correlated samples than what a slowly decaying autocorrelation function would indicate. Below, inference such as its mean, variance and bounds for a 95% credible set is given.

```
sigma_stats = data.frame("Mean" = mean(x$sigmas), "Variance" = var(x$sigmas),
                         "Lower.CS" = quantile(x$sigmas,0.025),
                         "Upper.CS" = quantile(x$sigmas,0.975))
print.data.frame(sigma_stats)
```

```
##           Mean    Variance   Lower.CS  Upper.CS
## 2.5% 0.1135489 0.000365171 0.08211743 0.1529612
```

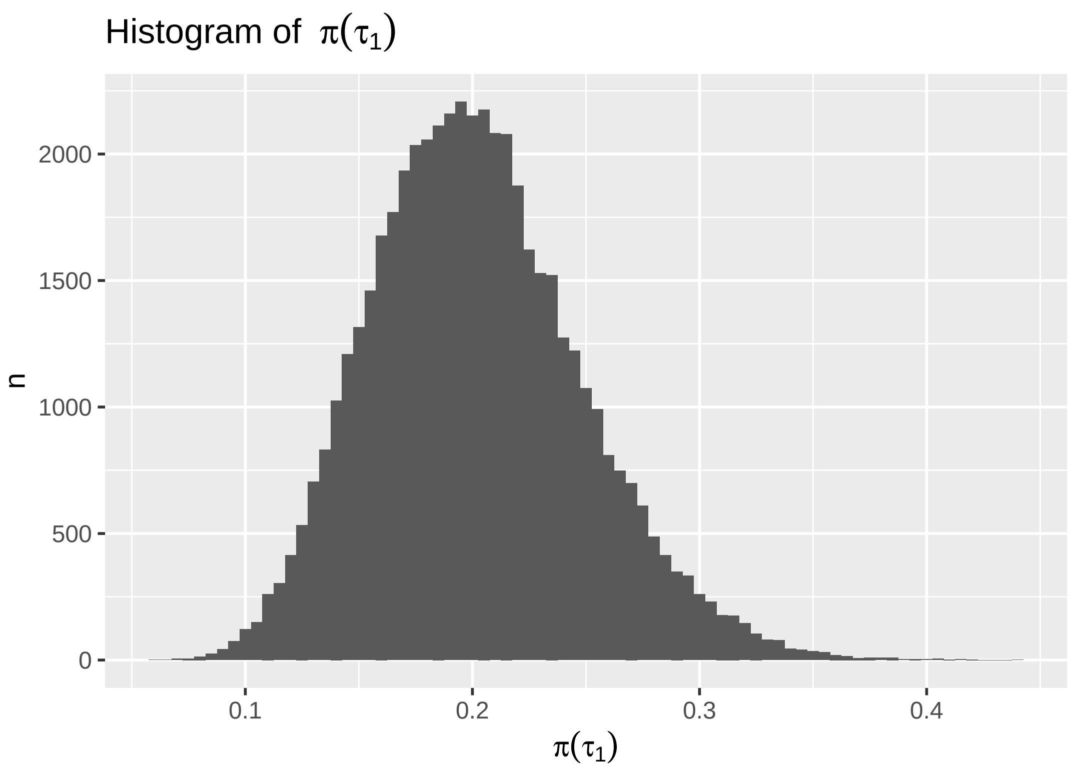
```
#tau1

pi1tsdf = data.frame("Pi1" = pidf$V1[1:5000], n = 1:5000)
(traceplot_pi1 = ggplot(pi1tsdf, aes(n, Pi1)) + geom_line() +
    labs(title = bquote("Traceplot of" ~ pi (tau[1])),
         caption = bquote("The figure shows a traceplot of the 5000 samples of " ~ pi (tau[1])),
         tag = "Figure 3.1") + ylab(bquote(pi(tau[1]))) +
    theme(plot.tag.position = "bottomleft"))
```



The figure shows a traceplot of the 5000 samples of $\pi(\tau_1)$

Figure 3.1

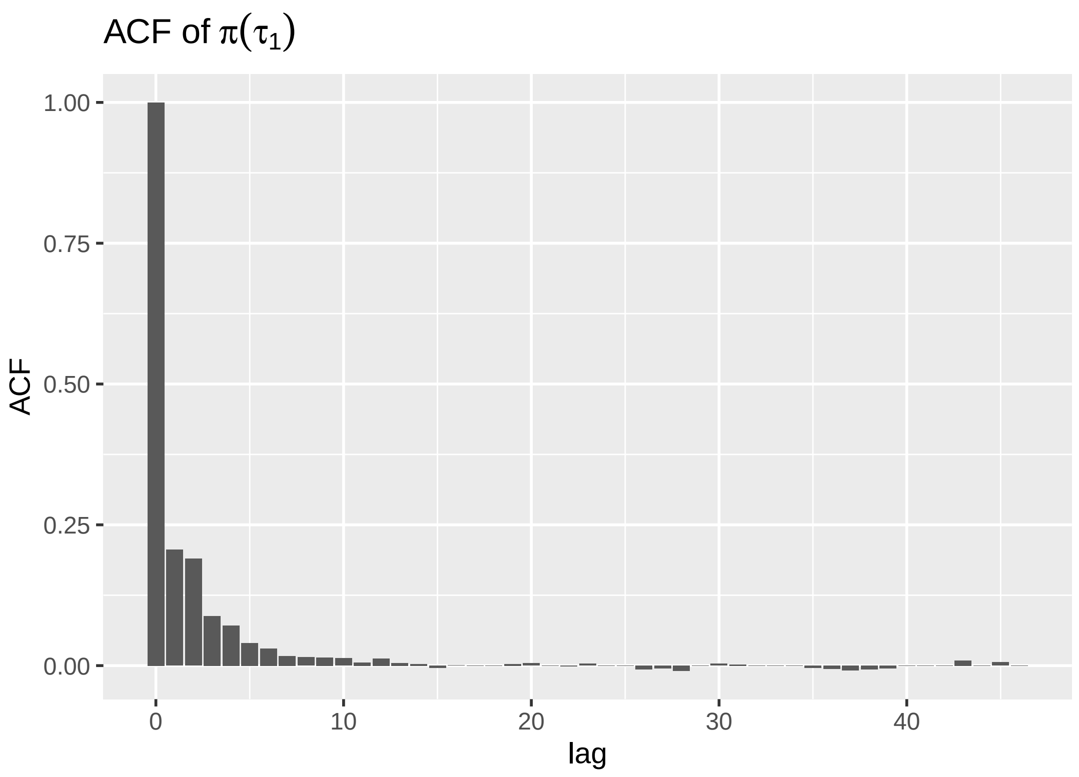
```r
pi1df = data.frame("Pi1" = pidf$V1, "n = 1:50000")
(histogram_pi1 <- ggplot(pi1df, aes(Pi1)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Histogram of " ~ pi(tau[1])),
       caption = bquote("The figure shows a histogram of " ~ pi(tau[1]) ~" for all iterations."),
       tag = "Figure 3.2") + xlab(bquote(pi(tau[1]))) + ylab("n") +
  theme(plot.tag.position = "bottomleft"))
```



The figure shows a histogram of $\pi(\tau_1)$ for all iterations.

Figure 3.2

```
bacf = acf(pi1df$Pi1, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(pi1_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[1])),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[1])),
       tag = "Figure 3.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```



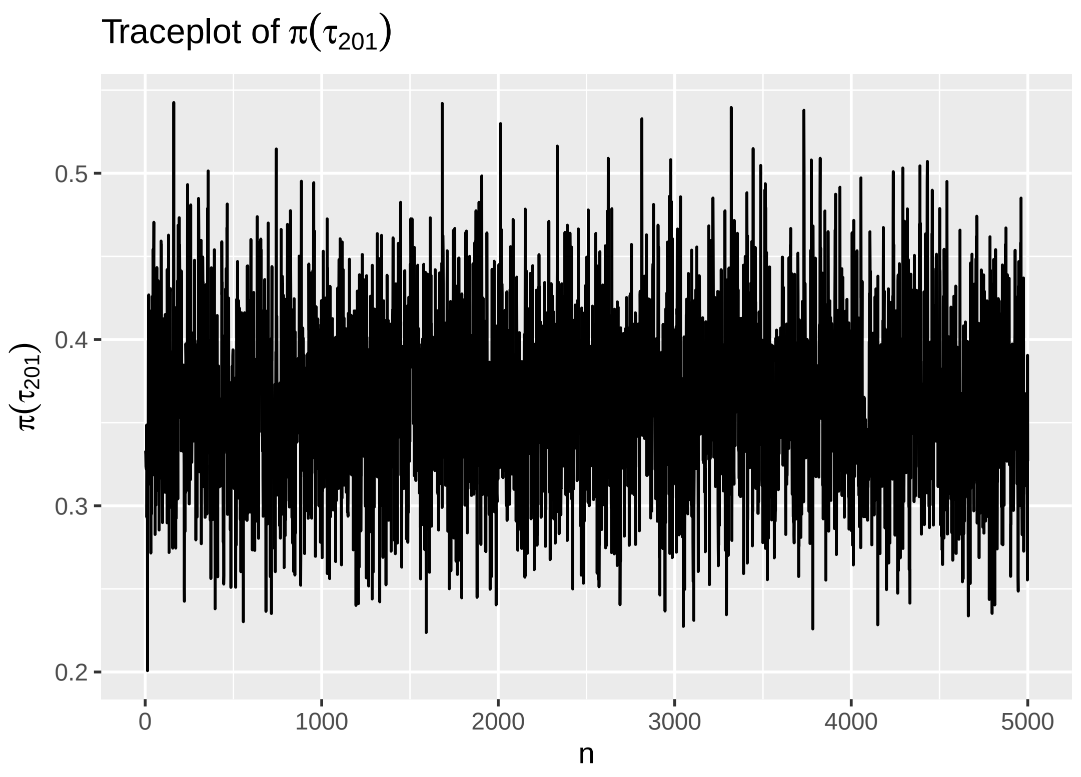
The figure shows the autocorrelation function of $\pi(\tau_1)$

Figure 3.3

```
pi1_stats = data.frame("Initial.rain.prob" = r$prob_rain[1], "Mean" = mean(x$pi[,1]),
                        "Variance" = var(x$pi[,1]),
                          "Lower.CSI" = quantile(x$pi[,1], 0.025),
                          "Upper.CS" = quantile(x$pi[,1],0.975))
print.data.frame(pi1_stats)
```

```
##      Initial.rain.prob      Mean    Variance Lower.CSI Upper.CS
## 2.5%         0.2051282 0.2018045 0.002152565  0.120637 0.302104
```

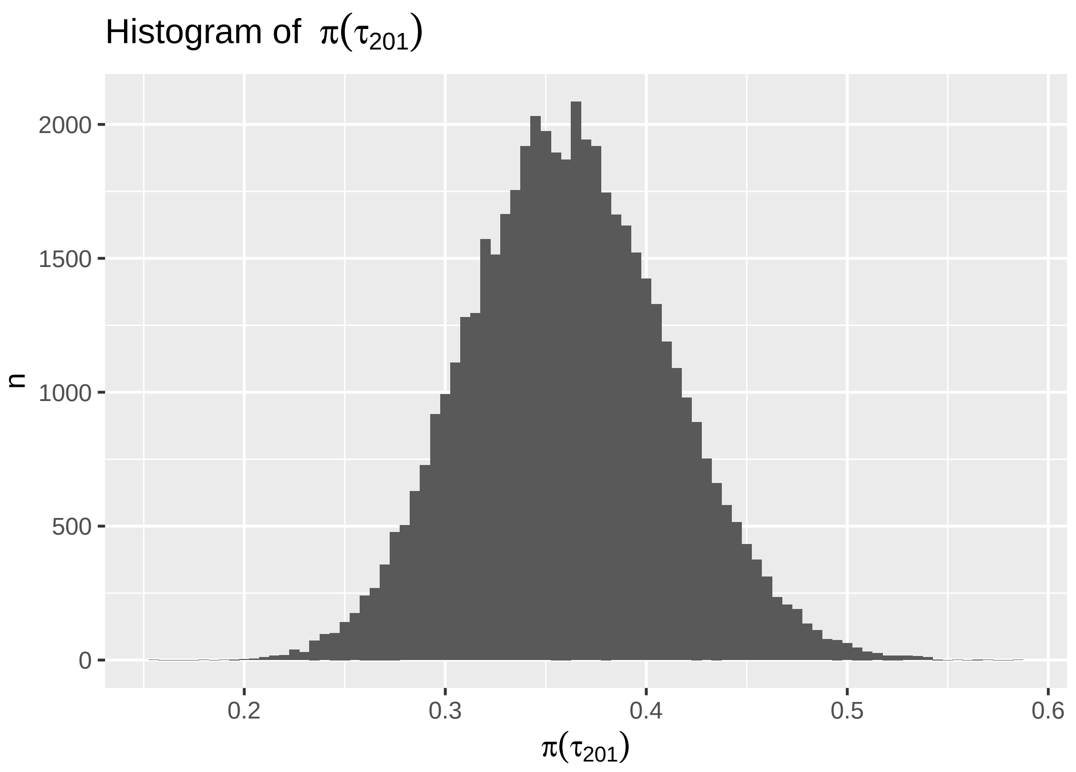
```
#tau201
pi201tsdf = data.frame("Pi201" = pidf$V201[1:5000], n = 1:5000)
(traceplot_pi1 = ggplot(pi201tsdf, aes(n, Pi201)) + geom_line() +
    labs(title = bquote("Traceplot of" ~ pi (tau[201])),
         caption = bquote("The figure shows a traceplot of the 5000 samples of " ~ pi (tau[201])),
         tag = "Figure 4.1") + ylab(bquote(pi(tau[201]))) +
    theme(plot.tag.position = "bottomleft"))
```



The figure shows a traceplot of the 5000 samples of $\pi(\tau_{201})$

Figure 4.1

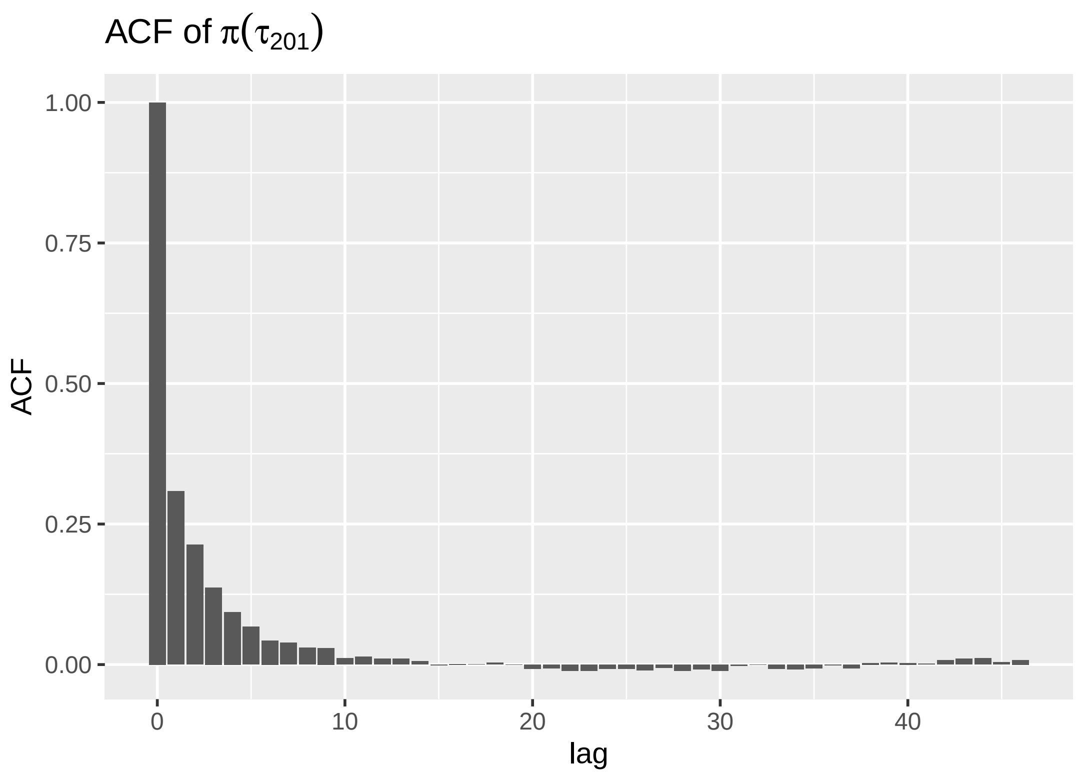
```
pi201df = data.frame("Pi201" = pidf$V201, "n" = 1:50000)
(histogram_pi201 <- ggplot(pi201df, aes(Pi201)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Histogram of " ~ pi(tau[201])),
       caption = bquote("The figure shows a histogram of " ~ pi(tau[201]) ~" for all iterations."),
       tag = "Figure 4.2") + xlab(bquote(pi(tau[201]))) + ylab("n") +
  theme(plot.tag.position = "bottomleft"))
```



The figure shows a histogram of $\pi(\tau_{201})$ for all iterations.

Figure 4.2

```
bacf = acf(pi201df$Pi201, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(pi201_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[201])),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[201])),
       tag = "Figure 4.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```



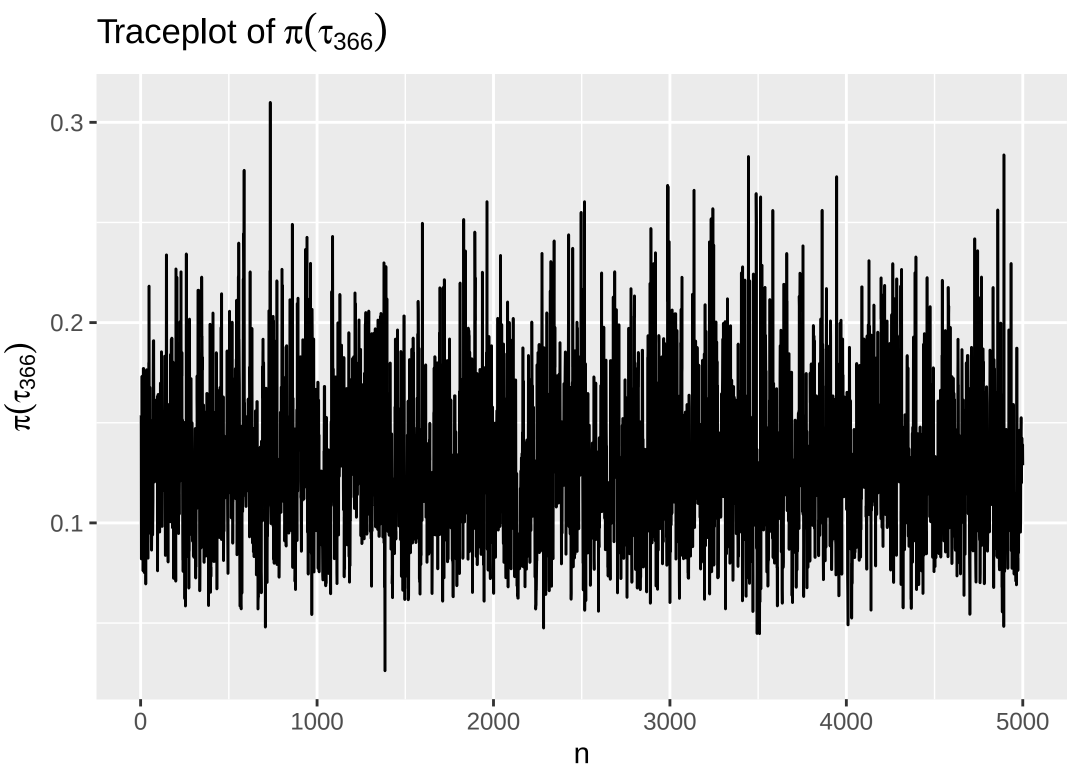
The figure shows the autocorrelation function of $\pi(\tau_{201})$

Figure 4.3

```
pi201_stats = data.frame("Initial.rain.prob" = r$prob_rain[201], "Mean" = mean(x$pi[,201]),
                          "Variance" = var(x$pi[,201]),
                         "Lower.CSI" = quantile(x$pi[,201], 0.025),
                         "Upper.CS" = quantile(x$pi[,201], 0.975))
print.data.frame(pi201_stats)
```

```
##      Initial.rain.prob      Mean    Variance Lower.CSI  Upper.CS
## 2.5%         0.3333333 0.3618009 0.002504617 0.2680925 0.4632057
```

```
#tau366

pi366tsdf = data.frame("Pi366" = pidf$V366[1:5000], n = 1:5000)
(traceplot_pi366 = ggplot(pi366tsdf, aes(n, Pi366)) + geom_line() +
    labs(title = bquote("Traceplot of" ~ pi (tau[366])),
         caption = bquote("The figure shows a traceplot of the 5000 samples of " ~ pi (tau[366])),
         tag = "Figure 5.1") + ylab(bquote(pi(tau[366]))) +
    theme(plot.tag.position = "bottomleft"))
```

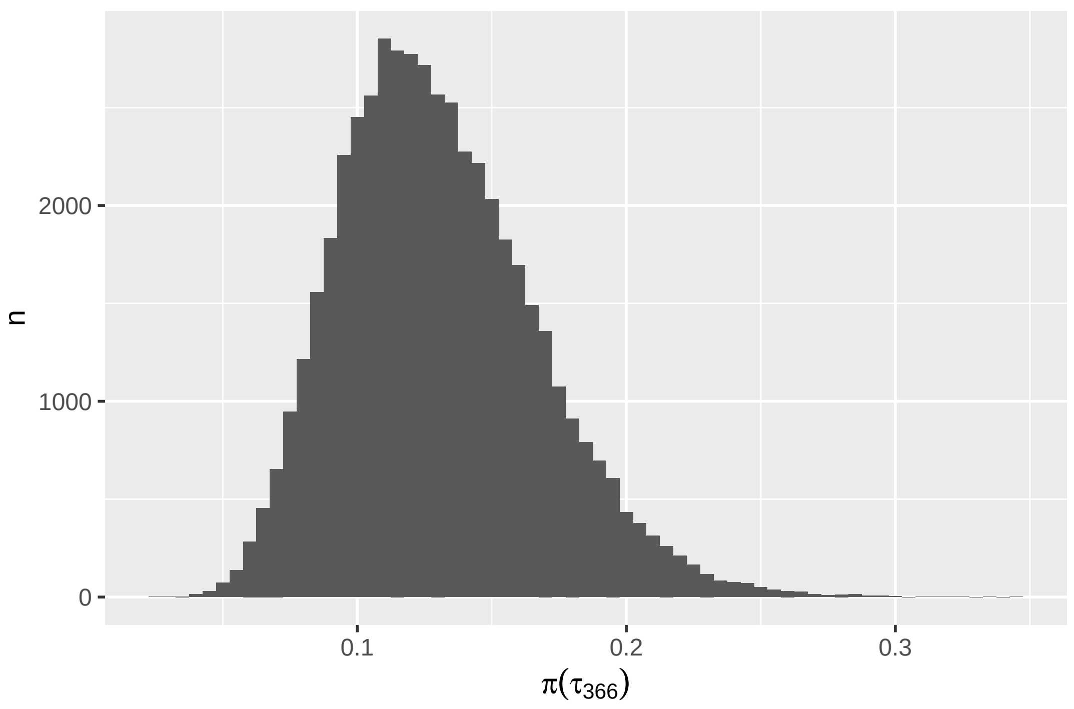


The figure shows a traceplot of the 5000 samples of $\pi(\tau_{366})$

Figure 5.1

```
pi366df = data.frame("Pi366" = pidf$V366, "n" = 1:50000)
(histogram_pi366 <- ggplot(pi366df, aes(Pi366)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Histogram of " ~ pi(tau[366])),
       caption = bquote("The figure shows a histogram of " ~ pi(tau[366]) ~" for all iterations."),
       tag = "Figure 5.2") + xlab(bquote(pi(tau[366]))) + ylab("n") +
  theme(plot.tag.position = "bottomleft"))
```

Histogram of $\pi(\tau_{366})$



The figure shows a histogram of $\pi(\tau_{366})$ for all iterations.

Figure 5.2

```r
bacf = acf(pi366df$Pi366, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(pi366_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[366])),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[366])),
       tag = "Figure 5.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```

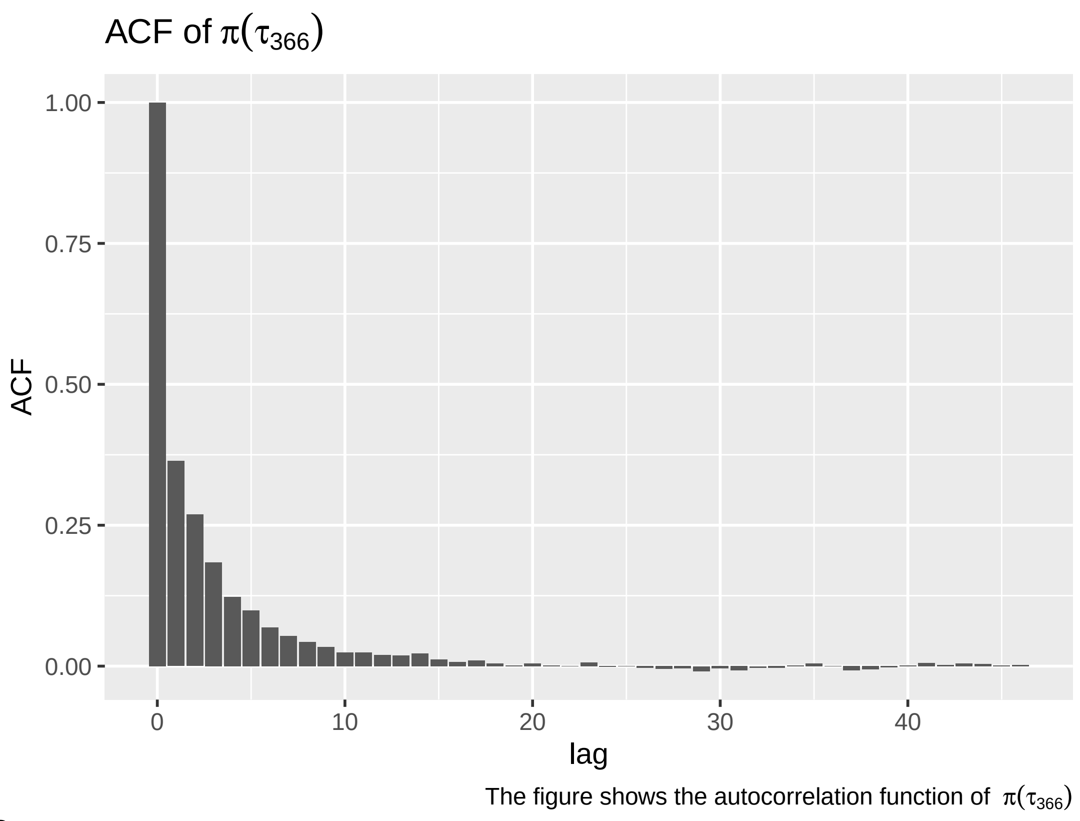


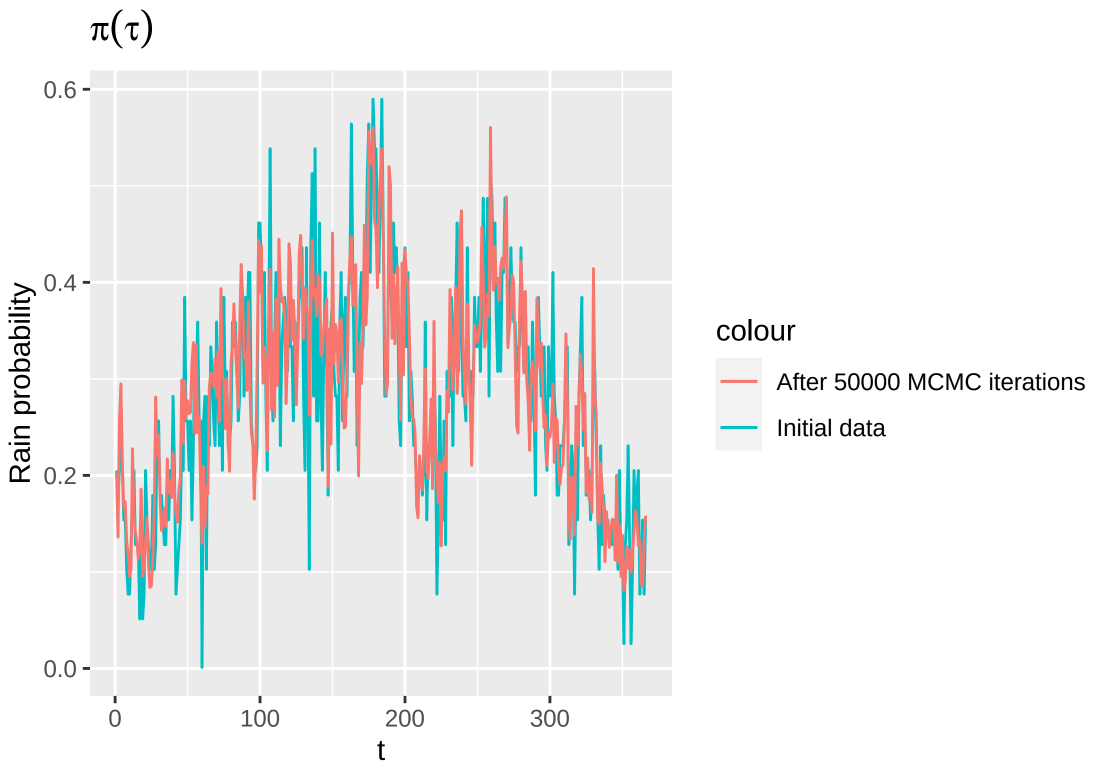
Figure 5.3

```r
pi366_stats = data.frame("Initial.rain.prob" = r$prob_rain[366], "Mean" = mean(x$pi[,366]), "Variance" =
                         "Lower.CSI" = quantile(x$pi[,366], 0.025),
                         "Upper.CS" = quantile(x$pi[,366], 0.975))
print.data.frame(pi366_stats)
```

```
##      Initial.rain.prob      Mean    Variance  Lower.CSI  Upper.CS
## 2.5%         0.1538462 0.1303504 0.001350299 0.06966295 0.2117744
```

All the values of $\pi(\tau_1)$, $\pi(\tau_{201})$ and $\pi(\tau_{366})$ converges relatively quick to its MCMC-approximated value as the traceplots in the figures 3.1, 4.1 and 5.1 show. The traceplots show that all the samples fluctuates around the means of approximately 0.2, 0.35 and 0.15, respectively. This is close to its initial value indicating that this is a stable value and that the algorithm converges early, which also coincides well with the fact that the value of σ_u^2 converges fast. Furthermore the histograms in the figures 3.2, 4.2 and 5.2 show values around estimated and initial values approximately normally distributed, and a relatively quickly decaying autocorrelation function from the figures 3.3, 4.3 and 5.3. This coincides well with she short or non-existing burn-in periods. We also see the inferences such as their means, variances and bounds for their 95% credible sets, which fits well with the traceplots and histograms, and the initial values of the probaibilities of rain at day 1, 201 and 366, respectively.

```r
pitaudf <- data.frame("t" = 1:366, "After.MCMC.50000" = x$pi[50000,], "Initial.data" = r$prob_rain)

(overallplot <- ggplot(pitaudf, aes(x = t)) +
    geom_line(aes(y = Initial.data, col = "Initial data")) +
    geom_line(aes(y = After.MCMC.50000, col = "After 50000 MCMC iterations")) +
    labs(title = bquote(pi(tau)),
         caption = bquote("Probability of rain" ~ pi(tau) ~ "from the initial data and after 50000 MCMC
         tag = "Figure 6") + ylab("Rain probability") +
    theme(plot.tag.position = "bottomleft"))
```



Probability of rain π(τ) from the initial data and after 50000 MCMC iterations
Figure 6

We confirm what we observed from the traceplots in the plot of $\pi(\tau)$ in Figure 6, seing that the MCMC-estimated rain probabilities has a similar trend as our initial data.

f)

We now want to implement a MCMC algorithm in the same way as above, except for a minor exception. Instead of updating every τ_t individually, we will update blocks of the vector τ of length M, i e. updating $\tau_{(a,b)} = (\tau_a, ..., \tau_b)^T$ from the conditional prior proposal $p(\tau_{(a,b)}|\tau_{-(a,b)}, \sigma_u^2)$.

Similarly as in problem e), due to sparse matrices, we do the following matrix multiplication simplifications.

$$\mathbf{Q}_{(a,b),-(a,b)}\tau_{-(a,b)} = \begin{cases} (0, ..., 0, -\tau_{b+1})^T & a = 1 \\ (-\tau_{a-1}, 0, ..., 0, -\tau_{b+1})^T & a > 1, b < T \\ (-\tau_{a-1}, 0, ..., 0)^T & b = T \end{cases}.$$

For the cases where $a = 1$ and $b = T$ we can simplify further.

$$-\mathbf{Q}^{-1}_{(a,b),(a,b)}\mathbf{Q}_{(a,b),-(a,b)}\tau_{-(a,b)} = \begin{cases} (\tau_{b+1}, ..., \tau_{b+1})^T & a = 1 \\ (\tau_{a-1}, ..., \tau_{a-1})^T & b = T \end{cases}.$$

With this taken into consideration, the function is implemented below.

```r
library(mvtnorm)
mcmc_RW2 = function(ntimes = 1000, M = 3)
{
  if (M == 1){
    return(mcmc_RW(ntimes = ntimes))
  }

  M_need = M        #saving M for later
  te = 366          #number of days
  Q = Q_def(te)     #366x366 precition matrix, excluding sigma_u

  # initial values
  init_pi = r$prob_rain #init probability of rain
  init_n = r$n.years #number of years for each day
  init_y = r$n.rain #number of days with rain more than 1mm
  init_tau = r$tau #initial tau value
  tau = matrix(nrow = ntimes, ncol = te) #creating a n x T matrix

  tau[1,] = init_tau #insering the initial tau into the matrix
  sigmas = vector(length = ntimes-1) #initialising a vector of sigmas

  alpha = 2 #given alpha value
  beta = 0.05 #given beta value
  count = 0 #counting the number of accepted values

  loopseq = seq(1,te,M) #creating loop for updating

  #precomputing
  Qinv_j1 = solve(Q[1:(1+M-1),1:(1+M-1)]) #Q inverse for j = 1
  Qinv_jm = solve(Q[2:(2+M-1),2:(2+M-1)]) #Q inverse for j = m
  #QinvQ_j1 = Qinv_j1 %*% Q[1:(1+M-1),-(1:(1+M-1))] #QAA mult QAB for j = 1
  QinvQ_j1 = matrix(0, nrow = M, ncol = te-M)
  QinvQ_j1[,1]=-1
  #QinvQ_jm = Qinv_jm %*% Q[2:(2+M-1),-(2:(2+M-1))] #QAA mult QAB for j = m
  l = loopseq[length(loopseq)]
  if((l+M-1) > te) { #In case te mod M != 0
    M = te-l+1
  }
  #Q inverse for j = te
  Qinv_jM = solve(Q[l:(l+M-1),l:(l+M-1)]) #Q inverse for j = M

  QinvQ_jM = Qinv_jM %*% Q[l:(l+M-1),-(l:(l+M-1))] #QAA mult. QAB for j = M

  M = M_need #resetting M to original value



  for(i in 2:ntimes) #starting the for loop
  {

    sigmas[i-1] = rinvgamma(1, shape = alpha + (te-1)/2, rate = 1/2*sum(diff(tau[i-1,])^2)+beta) #findi

    #loop for estimating each tau
    for(j in loopseq){
      if(j == 1){
        mu = -QinvQ_j1%*%tau[i-1,-(j:(j+M-1))]
        var = sigmas[i-1]*Qinv_j1
      }
      else if ((j+M-1) < te){
        #mu = -Qinv_jm %*% Q[j:(j+M-1),-(j:(j+M-1))] %*% tau[i-1,-(j:(j+M-1))]
        mult = rep(0, M)
        mult[1] = -1*tau[i-1,j-1]
        mult[M] = -1*tau[i-1,j+M]
        mu = -Qinv_jm %*% mult
        var = sigmas[i-1]*Qinv_jm
      }
      else { #In case te mod M != 0
        if((l+M-1) >= te) { #In case te mod M != 0
          M = te-l+1
        }
        mu = -QinvQ_jM %*% tau[i-1, -(j:(j+M-1))]
        var = sigmas[i-1]*Qinv_jM
      }

      temp = rmvnorm(1, mean = mu, sigma = var) #sample from mulitvariate normall

      temp_pi = 1/(1+exp(-temp))
      ratio = log_likelihood(init_n[j:(j+M-1)], init_y[j:(j+M-1)], temp_pi) - log_likelihood(init_n[j:(
      log_alpha = min(0, ratio)
      if(log(runif(1))< log_alpha){
        tau[i,j:(j+M-1)] = temp
        count = count + M #counting total number of times accepted
      }
      else
        tau[i,j:(j+M-1)] = tau[i-1,j:(j+M-1)]
    }

    M = M_need #In case te mod M != 0
  }
  acceptrate = count/(ntimes*(ceiling(te))) #general acceptance rate
  ret = list(tau = tau, rate = acceptrate, sigmas = sigmas, pi = 1/(1+exp(-tau)))
  return(ret)
}
start = proc.time()[3]
xM3 = mcmc_RW2(n=5000, M=3)
stop = proc.time()[3]

tM3 = stop - start

start = proc.time()[3]
xM5 = mcmc_RW2(n=5000, M=5)
stop = proc.time()[3]

tM5 = stop - start

start = proc.time()[3]
```

```
xM7 = mcmc_RW2(n=10000, M=7)
stop = proc.time()[3]

tM7 = stop - start


start = proc.time()[3]
xM10 = mcmc_RW2(n=10000, M=10)
stop = proc.time()[3]

tM10 = stop - start


start = proc.time()[3]
xM60 = mcmc_RW2(n=5000, M=60)
stop = proc.time()[3]

tM60 = stop - start
```

We get the acceptance ratios and process times for different block sizes below. `M3` represents the statistics when the blocks are of size $M = 3$, `M5` for block size $M = 5$ and so forth.

```
rate_time_blocks_df = data.frame("M3" = c(xM3$rate, tM3),
                                 "M5" = c(xM5$rate, tM5),
                                 "M7" = c(xM7$rate, tM7),
                                 "M10" = c(xM10$rate, tM10),
                                 "M60" = c(xM60$rate, tM10))
rownames(rate_time_blocks_df) = c("Acceptance.rate", "Process.time")
rate_time_blocks_df
```

```
##                          M3          M5          M7          M10          M60
## Acceptance.rate    0.5798902   0.4419951   0.3340284 6.163388e-03 8.557377e-04
## Process.time    3387.4200000 290.7500000 440.6800000 3.174600e+02 3.174600e+02
```

We observe that the acceptance probabilities decreases M increases. Too small acceptance probabilities indicate that almost none of the proposed steps are accepted, and the sampling is not of interest for any practical purpose. Take into account that the burn-in period are not removed before these magnitudes are computed. We introduce a threshold value of 0.01 for the acceptance probability to be relevant. Thus we go further with $M = \{3, 5, 7\}$ in our analysis. There also evidence for that the processing times decreases with increased block size M.

We now print some relevant statistics such as mean values and 95% credible sets for some different block sizes to assess the performance of the algorithm. For comparison we also print the corresponding statistics for the function from problem e).

```
Mstats <- data.frame("Without.blocks" = c(
      mean(x$pi[1,]),quantile(x$pi[1,], 0.025), quantile(x$pi[1,], 0.975),
      mean(x$pi[201,]),quantile(x$pi[201,], 0.025), quantile(x$pi[201,], 0.975),
      mean(x$pi[366,]),quantile(x$pi[366,], 0.025), quantile(x$pi[366,], 0.975)),
    "M3" = c(
      mean(xM3$pi[1,]),quantile(xM3$pi[1,], 0.025), quantile(xM3$pi[1,], 0.975),
      mean(xM3$pi[201,]),quantile(xM3$pi[201,], 0.025), quantile(xM3$pi[201,], 0.975),
      mean(xM3$pi[366,]),quantile(xM3$pi[366,], 0.025), quantile(xM3$pi[366,], 0.975)),
    "M5" = c(
      mean(xM5$pi[1,]),quantile(xM5$pi[1,], 0.025), quantile(xM5$pi[1,], 0.975),
      mean(xM5$pi[201,]),quantile(xM5$pi[201,], 0.025), quantile(xM5$pi[201,], 0.975),
      mean(xM5$pi[366,]),quantile(xM5$pi[366,], 0.025), quantile(xM5$pi[366,], 0.975)),
    "M7" = c(
      mean(xM7$pi[1,]),quantile(xM7$pi[1,], 0.025), quantile(xM7$pi[1,], 0.975),
      mean(xM7$pi[201,]),quantile(xM7$pi[201,], 0.025), quantile(xM7$pi[201,], 0.975),
      mean(xM7$pi[366,]),quantile(xM7$pi[366,], 0.025), quantile(xM7$pi[366,], 0.975)))

rownames(Mstats) = c("Mean.Pi(tau1)",
                     "Lower.CS.Pi(Tau1)", "Upper.CS.Pi(Tau1)",
                     "Mean.Pi(tau201)",
                     "Lower.CS.Pi(Tau201)", "Upper.CS.Pi(Tau201)",
                     "Mean.Pi(tau366)",
                     "Lower.CS.Pi(Tau366)", "Upper.CS.Pi(Tau366)")
Mstats
```

```
##                       Without.blocks         M3         M5         M7
## Mean.Pi(tau1)             0.28142350 0.28142350 0.28142350 0.28142350
## Lower.CS.Pi(Tau1)         0.07692308 0.07692308 0.07692308 0.07692308
## Upper.CS.Pi(Tau1)         0.51282051 0.51282051 0.51282051 0.51282051
## Mean.Pi(tau201)           0.27978743 0.28787558 0.27787210 0.28044385
## Lower.CS.Pi(Tau201)       0.10764340 0.10083164 0.08915782 0.07727751
## Upper.CS.Pi(Tau201)       0.49160490 0.47700723 0.51447003 0.51282051
## Mean.Pi(tau366)           0.28131181 0.28516006 0.28082515 0.27906399
## Lower.CS.Pi(Tau366)       0.10004274 0.10186823 0.08648490 0.07530088
## Upper.CS.Pi(Tau366)       0.47766756 0.48554566 0.52341200 0.51422750
```

We observe that all the sample means are similar for different block sizes. This is an indication of the code running as we could expect. Furthermore all the credible sets seem to be of approximatly the same size, independent of the block size.

The effective sample size (ESS) is a good measure of the performance of different MCMC samplers. The sample ESS can be defined by

$$\hat{\text{ESS}} = \frac{N}{\hat{\gamma}},$$

where the estimate of γ, $\hat{\gamma}$ is defined as

$$\hat{\gamma} = 1 + 2\sum_{k=1}^{K} \hat{\rho}(k),$$

and $\hat{\rho}(k)$ is the sample autocorrelation function at lag k. In our case K is the amount of lags considered in our ACF function. We want this to be as high as possible for a good MCMC sampler.

As all the means and variances for all considered block sizes seem to be relatively equal, we consider the processing times in combination with the ESS. As we want the highest ESS and the lowest processiong time, a reasonable diagnostic statistic is $\frac{\text{ESS}}{\text{Process time}}$. These magnitudes are computed below.

```
ESS <- function(pival){
  bacf = acf(pival, plot = F)
  bacfdf = with(bacf, data.frame(lag, acf))
  return(length(pival)/(1 + 2*sum(bacfdf$acf[2:length(bacf$acf)])))
}
```

```r
pt_ESS_df = data.frame("M1" = c(pt, ESS(x$pi[,201]), ESS(x$pi[,201])/pt),
                       "M3" = c(tM3, ESS(xM3$pi[,201]), ESS(xM3$pi[,201])/tM3),
                       "M5" = c(tM5, ESS(xM5$pi[,201]), ESS(xM5$pi[,201])/tM5),
                       "M7" = c(tM7, ESS(xM7$pi[,201]), ESS(xM7$pi[,201])/tM7))
rownames(pt_ESS_df) = c("Process.time", "ESS", "ESS/P.t")
pt_ESS_df
```

```
##                       M1           M3         M5         M7
## Process.time   528.39000 3387.4200000 290.750000 440.680000
## ESS          17378.29774  774.8627509 504.415687 527.981894
## ESS/P.t         32.88915    0.2287472   1.734878   1.198107
```

We observe that the processing time of $M = 1$ is slower than for $M = 3$, but it has a significantly higher ESS. Of the blocked samples, the block size $M = 3$ seems to give the best diagnostics we have decided to use. The samples of `M1` has a significantly better diagnostic than the blocked samples, the main reason being the significantly higher ESS.

Below we conider the traceplots, histograms and autocorrelation functions of the samples of `M3`, `M5` and `M7`. We choose $\pi(\tau_{201})$ as the probability to assess their performance.

```r
M3df <- data.frame("n" = 1:length(xM3$pi[,201]), "Pi201" = xM3$pi[,201])
(traceplot_M3 = ggplot(M3df, aes(n, Pi201)) + geom_line() +
  labs(title = bquote("Block update traceplot of" ~ pi(tau[201]) ~ "For M=3"),
       caption = bquote("Traceplot of " ~ pi(tau[201]) ~ "with block opdates for block length M = 3."),
       tag = "Figure 7.1") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```

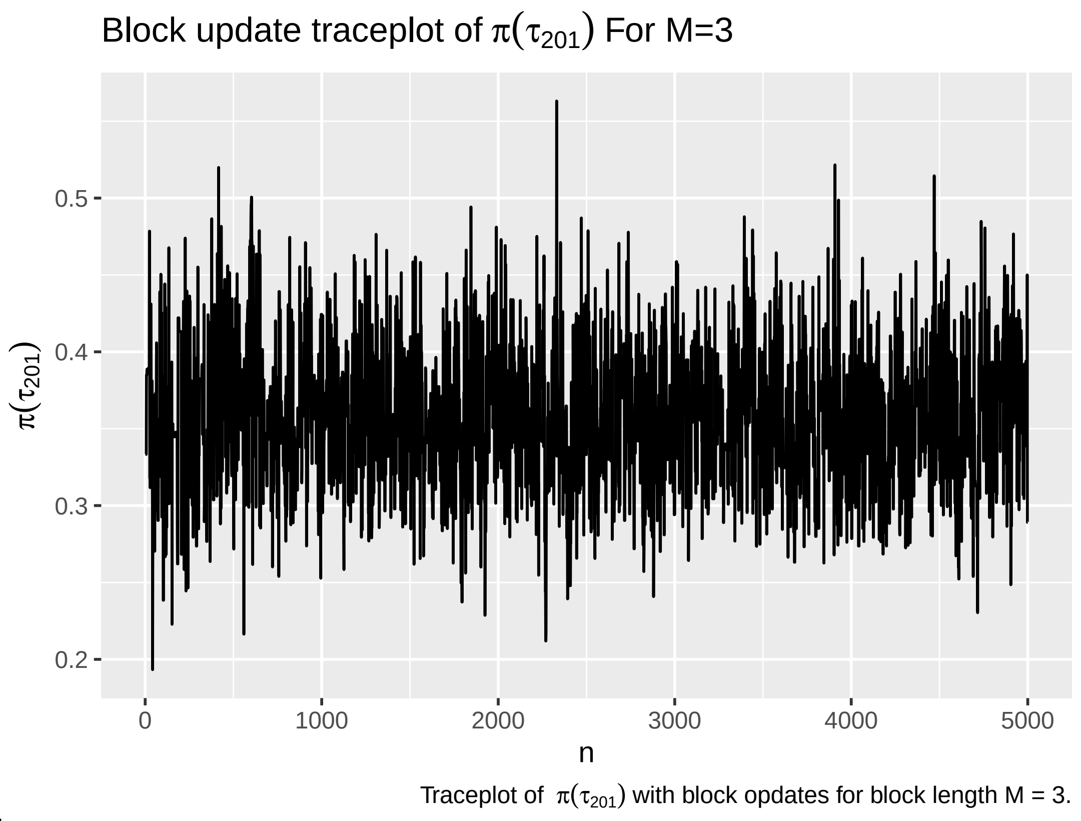


Figure 7.1

```
(histogram_M3 = ggplot(M3df, aes(Pi201)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Block update histogram of" ~ pi(tau[201]) ~ "For M=3"),
       caption = bquote("Histogram of " ~ pi(tau[201]) ~ "with block opdates for block length M = 3."),
       tag = "Figure 7.2") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```

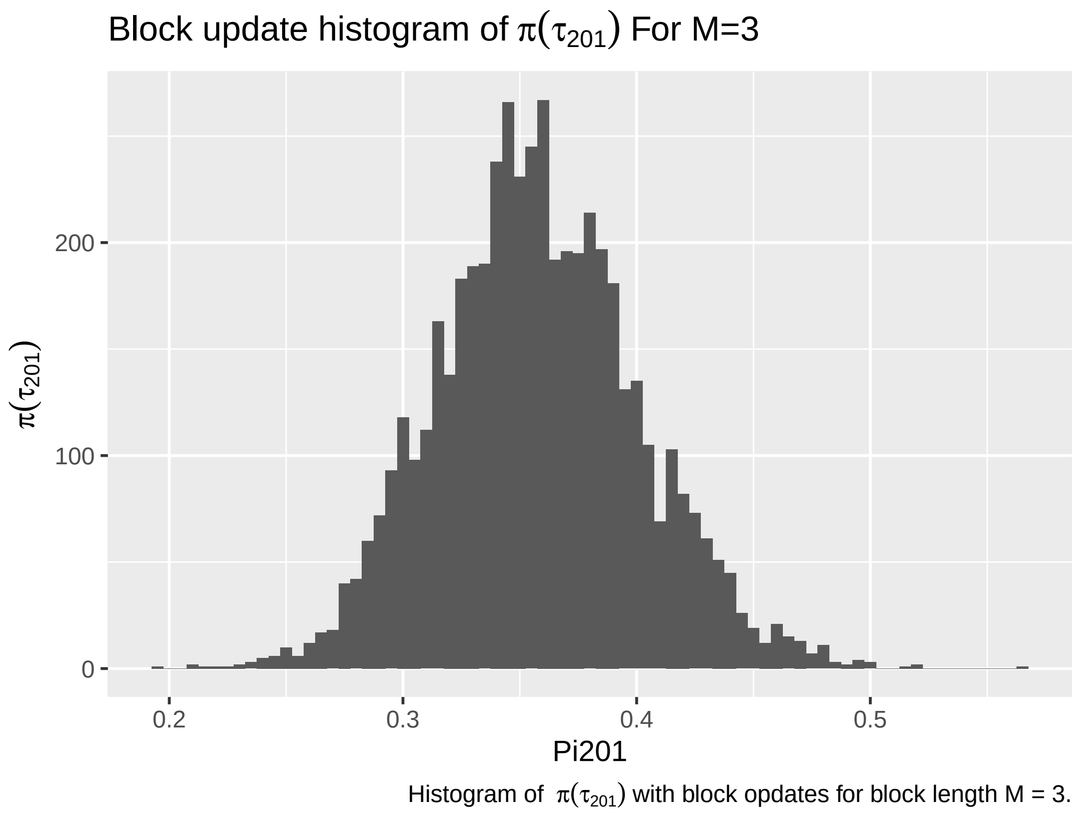
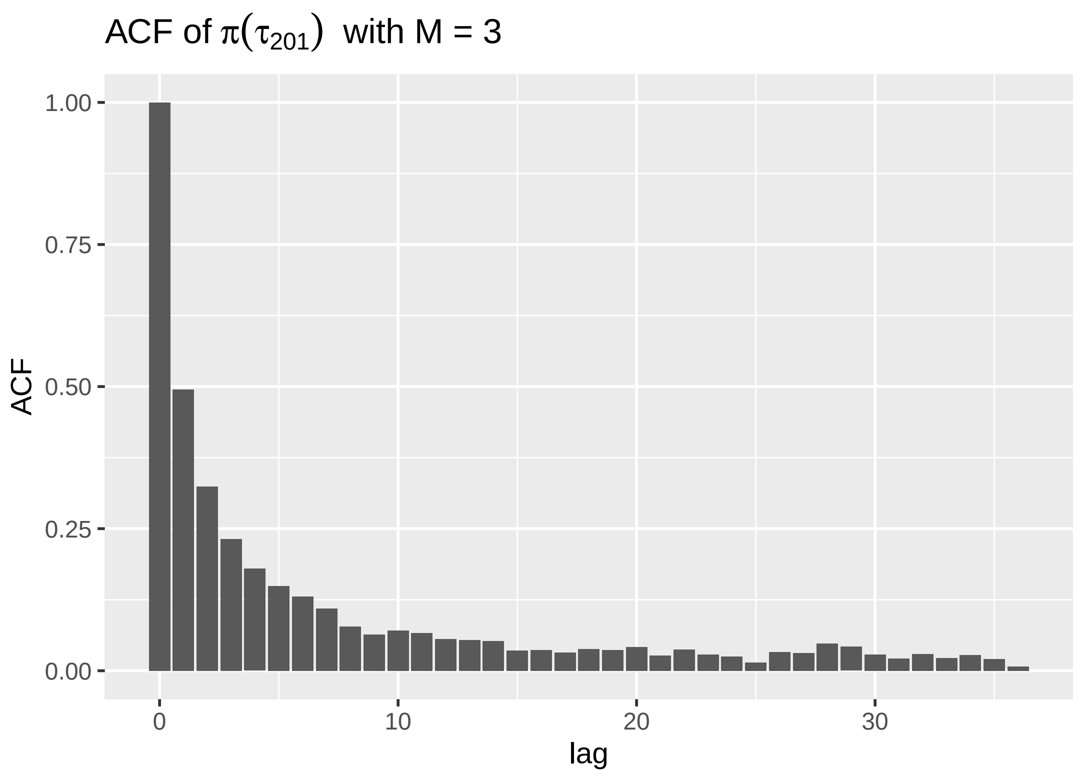


Figure 7.2

```
bacf = acf(M3df$Pi201, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(M3pi201_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[201]) ~" with M = 3"),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[201]) ~ "with M =
       tag = "Figure 7.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```

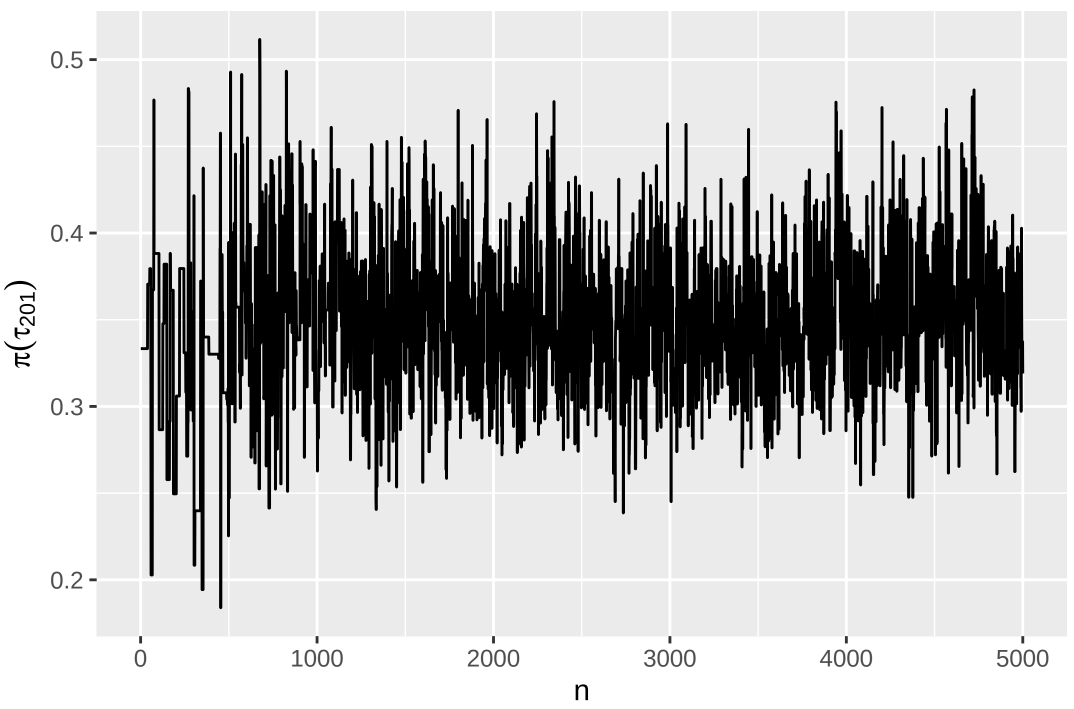


The figure shows the autocorrelation function of $\pi(\tau_{201})$ with M = 3 as block size.

Figure 7.3

```
### M=10

M5df <- data.frame("n" = 1:length(xM5$pi[,201]), "Pi201" = xM5$pi[,201])
(traceplot_M5 = ggplot(M5df, aes(n, Pi201)) + geom_line() +
  labs(title = bquote("Block update traceplot of" ~ pi(tau[201]) ~ "For M=5"),
       caption = bquote("Traceplot of " ~ pi(tau[201]) ~ "with block opdates for block length M = 5."),
       tag = "Figure 8.1") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```

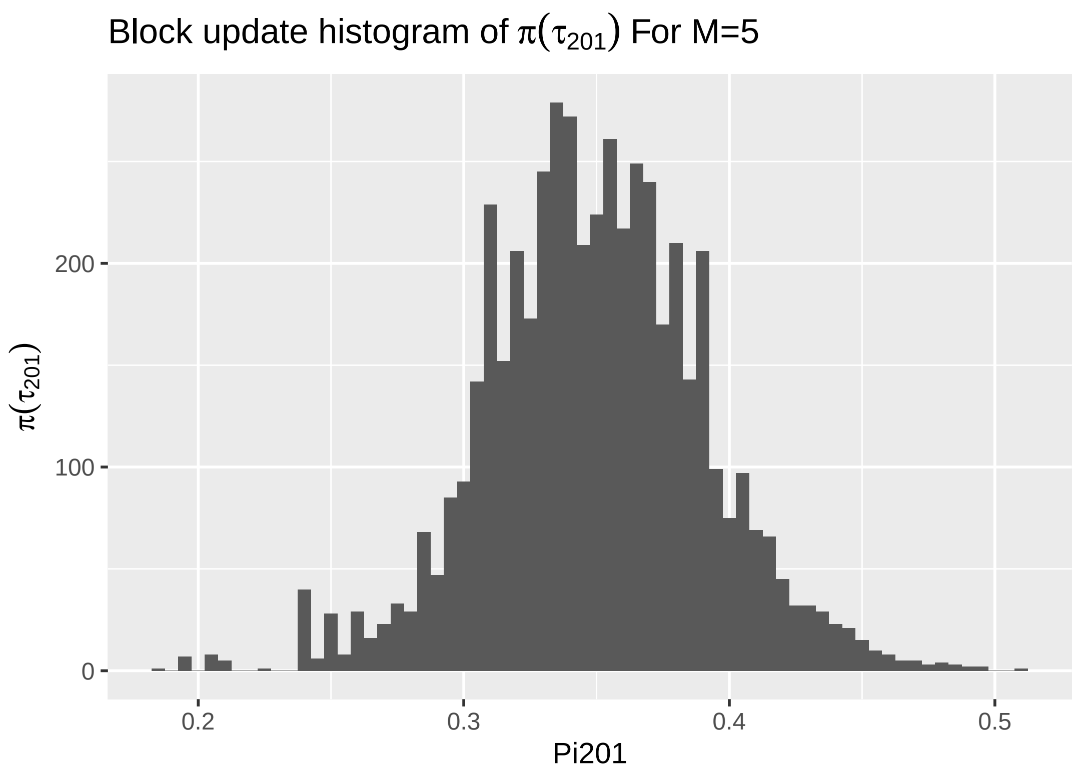
Block update traceplot of $\pi(\tau_{201})$ For M=5



Traceplot of $\pi(\tau_{201})$ with block opdates for block length M = 5.

Figure 8.1

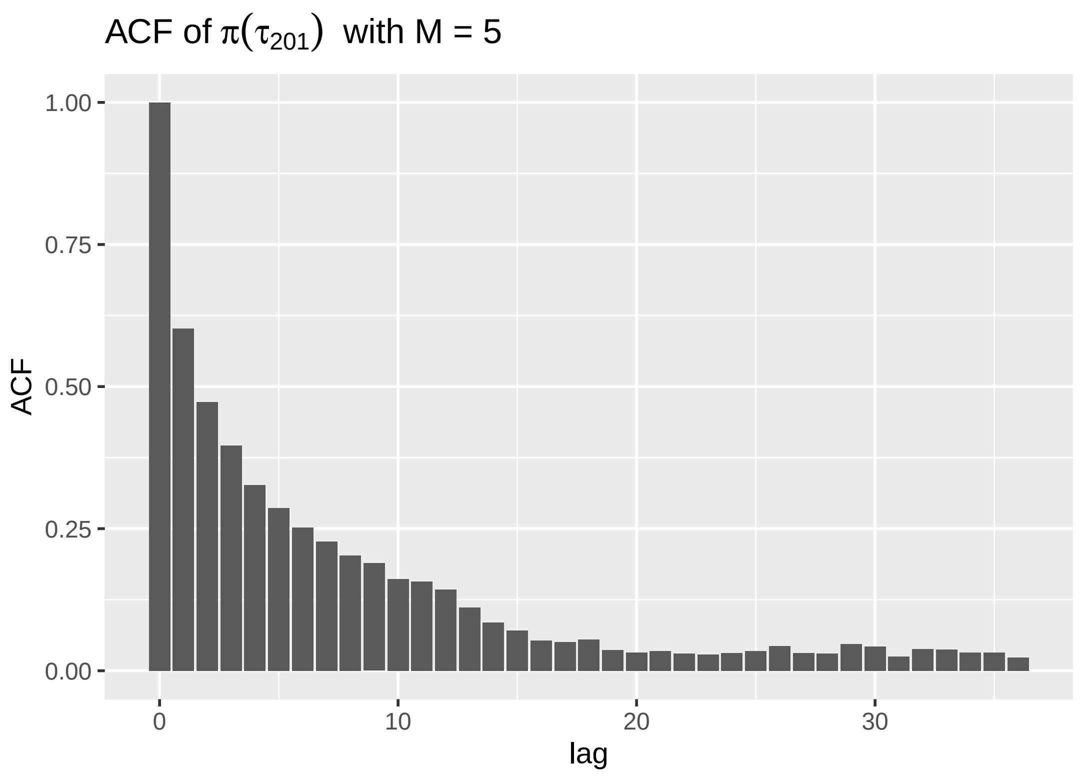
```
(histogram_M5 = ggplot(M5df, aes(Pi201)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Block update histogram of" ~ pi(tau[201]) ~ "For M=5"),
       caption = bquote("Histogram of " ~ pi(tau[201]) ~ "with block opdates for block length M = 5."),
       tag = "Figure 8.2") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```



Histogram of $\pi(\tau_{201})$ with block opdates for block length M = 5.

Figure 8.2

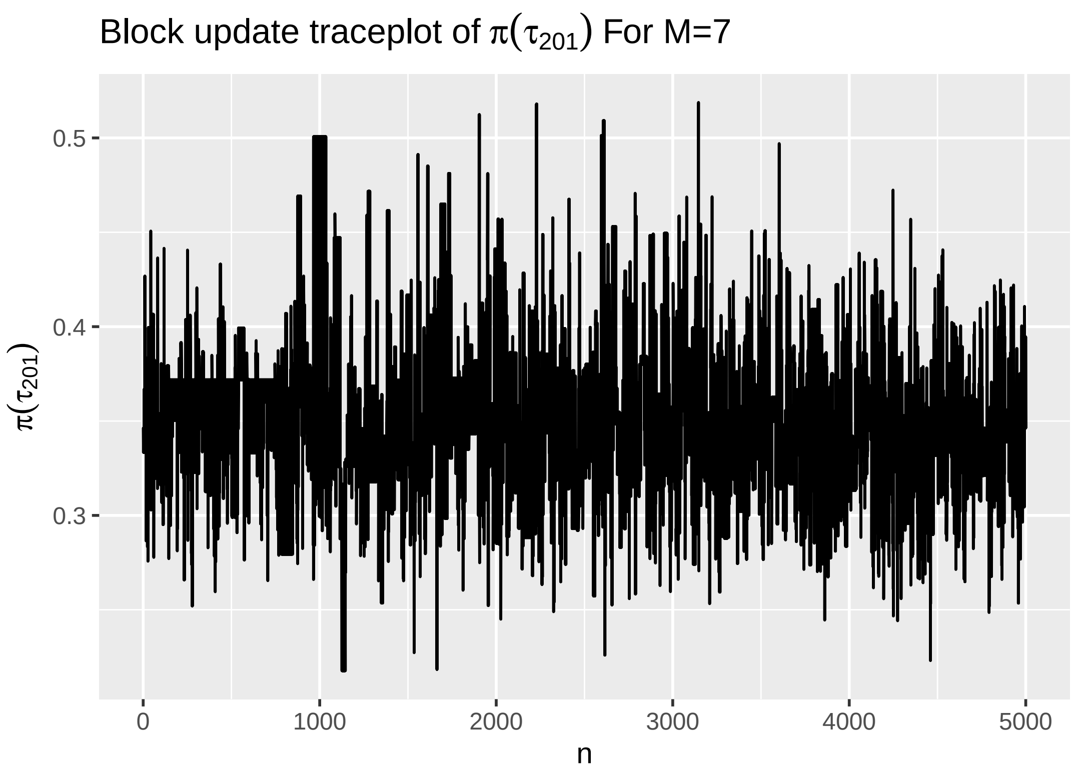
```
bacf = acf(M5df$Pi201, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(M5pi201_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[201]) ~" with M = 5"),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[201]) ~ "with M =
       tag = "Figure 8.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```



The figure shows the autocorrelation function of $\pi(\tau_{201})$ with M = 5 as block size.

Figure 8.3

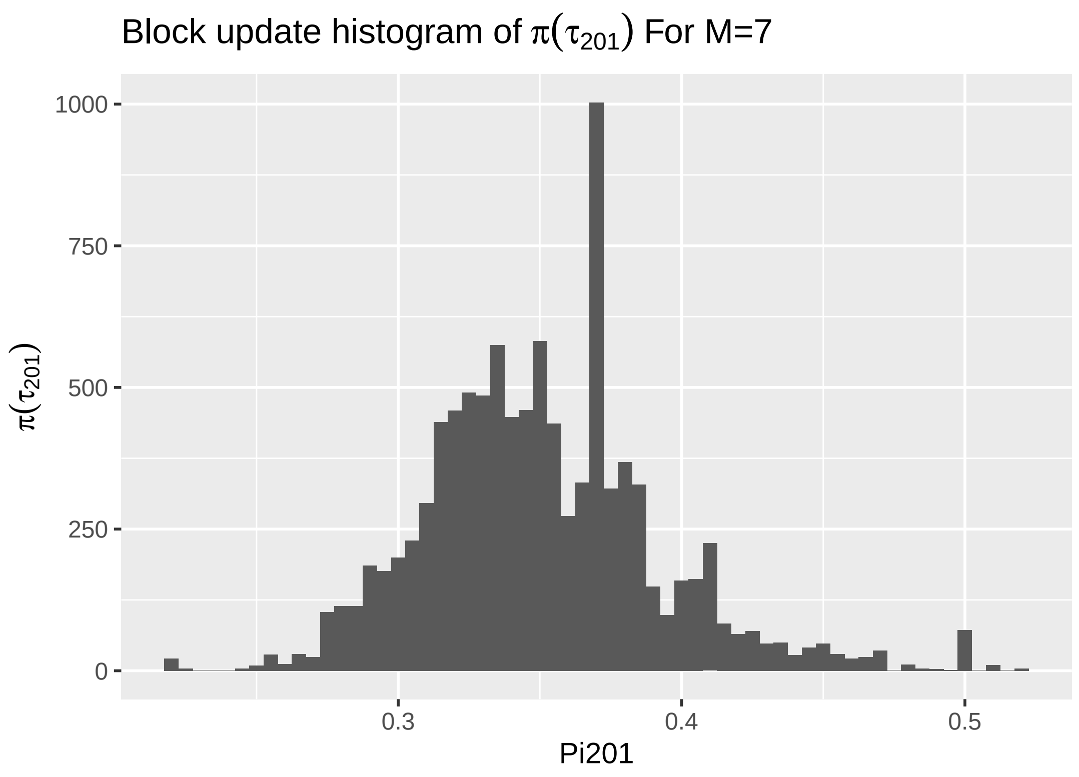
```
### M = 60
M7df <- data.frame("n" = 1:length(xM60$pi[,201]), "Pi201" = xM7$pi[,201])
(traceplot_M7 = ggplot(M7df, aes(n, Pi201)) + geom_line() +
  labs(title = bquote("Block update traceplot of" ~ pi(tau[201]) ~ "For M=7"),
       caption = bquote("Traceplot of " ~ pi(tau[201]) ~ "with block opdates for block length M = 7."),
       tag = "Figure 9.1") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```



Traceplot of $\pi(\tau_{201})$ with block opdates for block length M = 7.

Figure 9.1

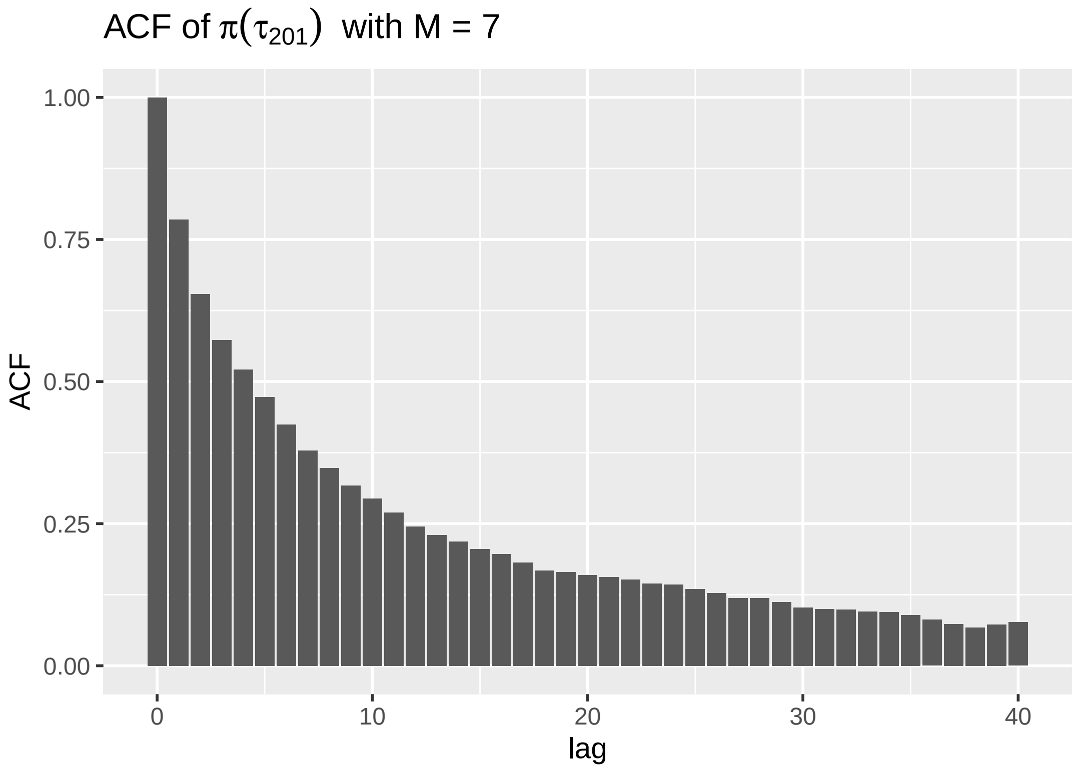
```
(histogram_M7 = ggplot(M7df, aes(Pi201)) + geom_histogram(binwidth = 0.005) +
  labs(title = bquote("Block update histogram of" ~ pi(tau[201]) ~ "For M=7"),
       caption = bquote("Histogram of " ~ pi(tau[201]) ~ "with block opdates for block length M = 7."),
       tag = "Figure 9.2") + ylab(bquote(pi(tau[201]))) + theme(plot.tag.position = "bottomleft"))
```



Histogram of $\pi(\tau_{201})$ with block opdates for block length M = 7.

Figure 9.2

```
bacf = acf(M7df$Pi201, plot = F)
bacfdf = with(bacf, data.frame(lag, acf))
(M7pi201_acf = ggplot(bacfdf, aes(lag, acf)) +
  geom_bar(stat = "identity", position = "identity") +
  labs(title = bquote("ACF of" ~ pi(tau[201]) ~" with M = 7"),
       caption = bquote("The figure shows the autocorrelation function of "  ~ pi(tau[201]) ~ "with M =
       tag = "Figure 9.3") + ylab("ACF") +
  theme(plot.tag.position = "bottomleft"))
```



The figure shows the autocorrelation function of $\pi(\tau_{201})$ with M = 7 as block size.

Figure 9.3

We see from the traceplots in the figures 7.1, 8.1 and 9.1 that as M increases, more samples are rejected. This can also be seen by the decreasing smoothness of the histogram in the figures 7.2, 8.2 and 9.2 as M increases. From the autocorelation functions of the figures 7.3, 8.3 and 9.3 we observe that the samples are more correlated for higher block size M. As the acceptance probabilities becomes lower for increasing M, the samples are less accepted and the same samples occur several times in a row. Because of this the correlation between samples become higher.

We conclude by saying that the MCMC RW(1) sampler without blocks gives the best performing MCMC sampler for this dataset with our implementation taken into consideration.

Problem 2

We will in this section consider the same dataset but use INLA rather than MCMC. We begin the section loading the INLA package into our environment.

```r
library(INLA)
```

a)

From the documentation (inla.doc("rw1")), it is clear that the precision matrix is on the form $\mathbf{R} = a\mathbf{Q}$ where the precision parameter a is represented as $\theta = \log a$, where again θ is considered the prior in the INLA model. In our case the precision matrix is defined as $\frac{1}{\sigma_u^2}\mathbf{Q}$ where σ_u^2 is inverse gamma distributed. Since $a = \sigma_u^{-2}$ we have that $a \sim \Gamma(\alpha, \beta)$ and thus that the prior has the distribution $\theta \sim \log \Gamma(\alpha, \beta)$.

We now fit the same model as in the previous problem using INLA.

```
init_t = proc.time()[3]
control.inla = list(strategy="simplified.laplace", int.strategy="ccd")
hyper = list(prec = list(prior = "loggamma", param = c(2,0.05)))
mod <- inla(n.rain ~ -1 + f(day, model="rw1", constr=FALSE), data=rain, Ntrials=n.years,
           control.compute=list(config = TRUE), family="binomial", verbose=TRUE,
           control.inla=control.inla)
finish_t = proc.time()[3]
proc_time = finish_t - init_t
```

Here we have used `ccd` integration rather than Laplace approximation and grid integration as a simplified Laplace approximation. We have removed the intercept by the `-1` option and by the `model = "rw1"` in the `f()` function we have fitted a RW(1) model.

```
proc_time
```

```
## elapsed
##    3.11
```

First of all, we observe a significantly lower running time for INLA than for MCMC, which is a lot more practical.

We now extract some summary statistics from our INLA model.

```
INLAdf = rbind(mod$summary.fitted.values[1,], mod$summary.fitted.values[201,],
               mod$summary.fitted.values[366,])
INLAdf = as.data.frame(INLAdf[,-c(4,6)])
rownames(INLAdf) = c(bquote(pi(tau[1])), bquote(pi(tau[201])), bquote(pi(tau[366])))
INLAdf
```

```
##                     mean         sd 0.025quant 0.975quant
## pi(tau[1])   0.1797841 0.02603099 0.13282753  0.2347736
## pi(tau[201]) 0.3282038 0.02582726 0.27924080  0.3805814
## pi(tau[366]) 0.1336175 0.02144721 0.09501459  0.1790164
```

For comparison we create a similar dataframe from our best performing MCMC sampler.

```
MCMCdf <- data.frame("mean" = c(mean(x$pi[,1]), mean(x$pi[,201]), mean(x$pi[,366])),
                     "sd" = c(sd(x$pi[,1]), sd(x$pi[,201]), sd(x$pi[,366])),
                     "0.025quant" = c(quantile(x$pi[,1], 0.025),
                                      quantile(x$pi[,201], 0.025),
                                      quantile(x$pi[,366], 0.025)),
                     "0.975quant" = c(quantile(x$pi[,1], 0.925),
                                      quantile(x$pi[,201], 0.925),
                                      quantile(x$pi[,366], 0.925)))
rownames(MCMCdf) = c(bquote(pi(tau[1])), bquote(pi(tau[201])), bquote(pi(tau[366])))
MCMCdf
```

```
##                     mean         sd X0.025quant X0.975quant
## pi(tau[1])   0.2018045 0.04639574  0.12063702   0.2721284
## pi(tau[201]) 0.3618009 0.05004615  0.26809247   0.4354736
## pi(tau[366]) 0.1303504 0.03674642  0.06966295   0.1867779
```

We see that the statistics from the different models differs a little bit, but they are all in the same numerical neighborhood. A small difference is reasonable as for the MCMC sampler the burn-in samples are not removed and INLA only gives an approximate estimation of the true values. That being said, the means of the different samplers still has errors of too big magnitude to be completely satisfactory. Bith the standard deviations and the quantiles looks reasonable.

b

We now want to assess the robustness of the result to the `control.inla` input. We assess this by checking the same results removing the `control. inla` input.

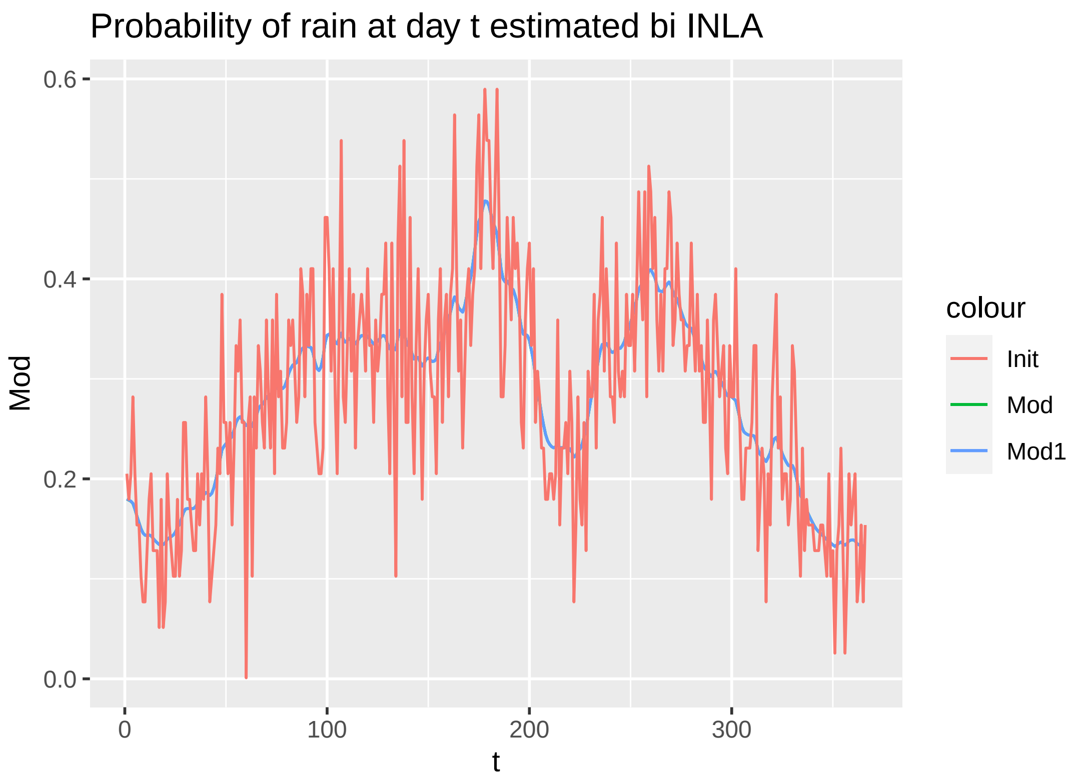
```r
init_t1 = proc.time()[3]
hyper = list(prec = list(prior = "loggamma", param = c(2,0.05)))
mod1 <- inla(n.rain ~ -1 + f(day, model="rw1", constr=FALSE), data=rain, Ntrials=n.years,
           control.compute=list(config = TRUE), family="binomial", verbose=TRUE)
finish_t1= proc.time()[3]
```

Again, we create a dataframe to assess the results.

```r
INLAdf1 = rbind(mod1$summary.fitted.values[1,], mod1$summary.fitted.values[201,],
               mod1$summary.fitted.values[366,])
INLAdf1 = as.data.frame(INLAdf1[,-c(4,6)])
rownames(INLAdf1) = c(bquote(pi(tau[1])), bquote(pi(tau[201])), bquote(pi(tau[366])))
INLAdf1
```

```
##                    mean         sd 0.025quant 0.975quant
## pi(tau[1])   0.1798973 0.02623352 0.13280760  0.2355735
## pi(tau[201]) 0.3283778 0.02598426 0.27931431  0.3813310
## pi(tau[366]) 0.1336308 0.02151050 0.09484382  0.1791429
```

```r
ggdf = data.frame("t" = 1:366, "Mod" = mod$summary.fitted.values$mean, "Mod1" = mod1$summary.fitted.valu

(INLAplot = ggplot(ggdf, aes(x=t)) + geom_line(aes(y = Mod, col = "Mod")) +
  geom_line(aes(y = Mod1, col = "Mod1")) + geom_line(aes(y = Init, col = "Init")) +
    labs(title = "Probability of rain at day t estimated bi INLA",
         caption = "Probability of rain at day t using inla with two different control.inla inputs and 
         tag = "Figure 10") + theme(plot.tag.position = "bottomleft"))
```



Probability of rain at day t using inla with two different control.inla inputs and initial data.

Figure 10

We observe that the values are essentially the same. Looking at figure 10, the lines of the different models are overlapping. This indicates that the results to the two `control.inla` inputs of `ccd` simplified Laplace integration and proper Laplace integration.

```
(time1 = finish_t1 - init_t1)
```

```
## elapsed
##    2.34
```

However the running time stays low for both `control.inla` inputs.

c)

We will now consider a corresponding model with an automatically generated intercept. Thus we remove the `-1` entry in the definition of the function and create some plots of the output in the following code.

```
mod2 <- inla(n.rain ~ f(day, model="rw1", constr=TRUE),
             data=rain, Ntrials=n.years, control.compute=list(config = TRUE),
             family="binomial", verbose=TRUE, control.inla=control.inla)

INLAdf2 = rbind(mod2$summary.fitted.values[1,], mod2$summary.fitted.values[201,],
               mod2$summary.fitted.values[366,])
INLAdf2 = as.data.frame(INLAdf2[,-c(4,6)])
rownames(INLAdf2) = c(bquote(pi(tau[1])), bquote(pi(tau[201])), bquote(pi(tau[366])))
INLAdf2
```

```
##                      mean         sd 0.025quant 0.975quant
## pi(tau[1])    0.1797852 0.02603067 0.13282914  0.2347740
## pi(tau[201]) 0.3282028 0.02582685 0.27924060  0.3805796
## pi(tau[366]) 0.1336199 0.02144711 0.09501707  0.1790185
```

We observe that the prediction by the model including the intercept term is essentially the same as for the two previous considered INLA models. This yelds evidence that the intercept term seems to be mathematically indifferent.

This is backed up mathematically. The model we have been considering throughout the excercise is defined by

$$y_t|\tau_t \sim \text{Bin}(n_t, \pi(\tau_t)),$$

where

$$\pi(\tau_t) = \frac{\exp(\tau_t)}{1+\exp(\tau_t)} = \frac{1}{1+\exp(-\tau_t)}.$$

We now define a new variable κ including an automatical intercept β_0, such that

$$\kappa_t = \beta_0 + \tau_t.$$

Now, our new model is defined in the same way as above substituting τ_t by κ_t such that

$$y_t|\kappa_t \sim \text{Bin}(n_t, \pi(\kappa_t)),$$

where

$$\pi(\kappa_t) = \frac{\exp(\kappa_t)}{1+\exp(\kappa_t)} = \frac{1}{1+\exp(-\kappa_t)}.$$

The prior on β_0 is default, we have a loggamma$(\alpha, beta)$ prior on θ. As `constr = TRUE`, we have a sum-to-zero constraint such that

$$\sum_{t=1}^{T} \tau_t = 0.$$

Thus the model is not mathematically significantly different from the earlier models, as we also obtained from the summary outputs.

Conclusion

We have throughout this report derived analytic expressions for a hierarchical bayesian model, implemented MCMC samplers with and without block samplings, and used INLA to obtain corresponding inference to the same model. We have observed that the MCMC samplers have been difficult to implement with several time consuming calculations, and varying acceptance rates. Its results are however reasonable and in the neighborhood of the INLA estimates, ensuring a satisfactory precision of both the models. When our purpose is to obtain inference on such datasets as the Tokyo rainfall data, the downside of the running time of the MCMC sampler is too significant to perfer it over INLA, which would be the method to perfer for future similar problems.