Project 2

Computer Intensive Statistical Methods

Erling Fause Steen og Christian Oppegård Moen

08 03 2022

Contents

Introduction

The Tokyo rainfall dataset contain the amount of rainfall for each of the 366 days (including February 29.) for several years. We will consider a portion of this dataset, specifically from $1951 - 1989$, such that for each day $t \neq 60$ we have $n_t = 39$ observations and for $t = 60$, February 39, we have $n_t = 10$ observation. For each day we have the response $y_t = 0, 1, 2, ..., n_t$ being the amount of times the rainfall exceeded 1mm over the given period, given by

$$y_t|\tau_t \sim \text{Bin}(n_t, \pi(\tau_t)), \quad \pi(\tau_t) = \frac{\exp(\tau_t)}{1 + \exp(\tau_t)} = \frac{1}{1 + \exp(-\tau_t)}. \tag{1}$$

Here, $\pi(\tau_t)$ is the probability of rainfall exceeding 1mm and τ_t is the logit probability of exceedence. For this project we assume conditional independence among the $y_t|\tau_t \ \forall t = 1, 2, ..., 366$.

We will apply a Bayesian hierarchical model to the dataset, using a random walk of order 1 (RW1) to model the trend. For the model we will implement a Markov chain Monte Carlo (MCMC) sampler for the posterior using Metropolis-Hastings (MH) and Gibbs steps for specific parameters. Then we will investigate the accuracy and computational speed of the implementation compared to the built in method `INLA` in R.

Problem 1

a) Display

In Figure 1 we see the number of times the rainfall has exceeded 1mm. There seem to be fewer days in the start of the year and in the end of the year with an amount of rainfall exceeding 1 mm. This is in January and December. The number of days steadily increases until the begining of the summer which seems to be the period with the most days with an amount of rainfall over 1 mm. Then, the amount of days decreases during july and august before increasing during the autumn. There also seem to be fluctuations on a daily basis from the just mentiod trend in the data. The red dot is the observation for February 29.

```
load("./rain.rda")
day     = rain$day
n.rain  = rain$n.rain
n.years = rain$n.years
##Plotting the data
ggplot(data=rain, mapping=aes(x=day, y=n.rain)) +
  geom_line() +
  xlab("Day") +
  ylab("Number of days with more than 1mm rain") +
  geom_point(aes(x=day[60], y=n.rain[60]), colour="red")
```

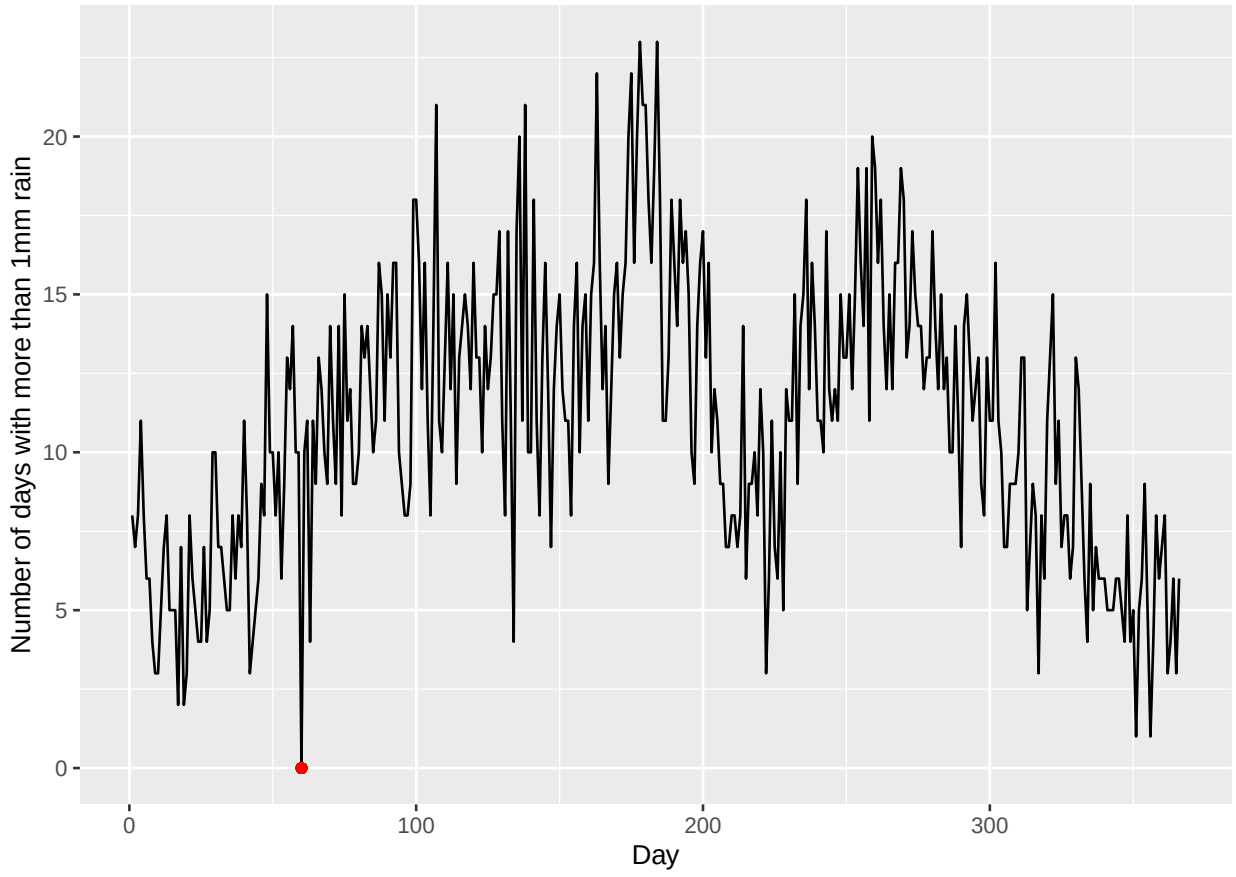


Figure 1: The Tokyo Rainfall dataset

b) Likelihood

The likelihood of Equation (1) is given by bc of conditional indep

$$\begin{aligned} L(\pi(\boldsymbol{\tau})) &= \prod_{i=1}^{T} \begin{pmatrix} n_t \\ y_t \end{pmatrix} \pi(\tau_t)^{y_t}(1-\pi(\tau_t)^{n_t-y_t} \\ &\propto \prod_{i=1}^{T} \pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{n_t-y_t} \\ &= \prod_{t=1}^{T} \left(\frac{\exp(\tau_t)}{1+\exp(\tau_t)}\right)^{y_t} \left(1-\frac{\exp(\tau_t)}{1+\exp(\tau_t)}\right)^{n_t-y_t}, \end{aligned}$$

where $\boldsymbol{\tau} = (\tau_1, ..., \tau_T)$, $y_t = 1, 2, ..., 39$ and $n_t = 39$ for $t \neq 60$, and $y_t = 1, 2, ..., 10$ and $n_t = 10$ for $t \neq 60$.

c) Posterior

As briefly mentioned in the introduction we need the posterior $P(\sigma^2|\tau, y)$ for the Gibbs step in our implementation, given by

$$\begin{aligned} P(\sigma^2|\boldsymbol{\tau}, \boldsymbol{y}) &= \frac{P(\sigma_u^2, \boldsymbol{\tau}, \boldsymbol{y})}{P(\boldsymbol{\tau}, \boldsymbol{y})} \\ &\propto P(\boldsymbol{y}|\sigma_u^2, \boldsymbol{\tau}) P(\sigma_u^2, \boldsymbol{\tau}) \\ &= P(\boldsymbol{y}|\sigma_u^2, \boldsymbol{\tau}) P(\tau|\sigma_u^2) P(\sigma_u^2), \end{aligned}$$

where $\boldsymbol{y} = (y_1, ..., y_T)^T$, $\boldsymbol{\tau} = (\tau_1, ..., \tau_T)$ and $P(\boldsymbol{y}|\sigma_u^2, \boldsymbol{\tau}) = L(\pi(\boldsymbol{\tau}))$. Based on model assumptions mentioned in the introduction, we have $\tau_t \sim \tau_{t-1} + u_t$ for $u_t \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2)$ so that

$$p(\boldsymbol{\tau}|\sigma_u^2) = \prod_{t=2}^{T} \frac{1}{\sigma_u} \exp\left\{-\frac{1}{2\sigma_u^2}(\tau_t - \tau_{t-1})^2\right\}.$$

We place an inverse gamma prior (IG) on σ_u^2 given by

$$p(\sigma_u^2) = \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} exp\left\{-\frac{\beta}{\sigma_u^2}\right\}.$$

Then, the posterior is

$$\begin{aligned} P(\sigma^2|\boldsymbol{\tau}, \boldsymbol{y}) &= \underbrace{\prod_{t=1}^{T} \left(\frac{\exp(\tau_t)}{1+\exp(\tau_t)}\right)^{y_t} \left(1-\frac{\exp(\tau_t)}{1+\exp(\tau_t)}\right)^{n_t-y_t}}_{\text{Constant w.r.t. } \sigma^2} \cdot \\ &\quad \prod_{t=1}^{T} \frac{1}{\sigma_u} \exp\left\{-\frac{1}{2\sigma_u^2}(\tau_t - \tau_{t-1})^2\right\} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} exp\left\{-\frac{\beta}{\sigma_u^2}\right\} \\ &\propto \prod_{t=1}^{T} \frac{1}{\sigma_u} \exp\left\{-\frac{1}{2\sigma_u^2}(\tau_t - \tau_{t-1})^2\right\} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} exp\left\{-\frac{\beta}{\sigma_u^2}\right\} \\ &= \frac{1}{\sigma_u^{T-1}} exp\left\{-\frac{1}{2\sigma_u^2}\boldsymbol{\tau}\boldsymbol{Q}\boldsymbol{\tau}\right\} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} exp\left\{-\frac{\beta}{\sigma_u^2}\right\} \\ &\propto \left(\frac{1}{\sigma_u^2}\right)^{\alpha+\frac{T-1}{2}+1} exp\left\{-\frac{1}{\sigma_u^2}\left(\frac{1}{2}\boldsymbol{\tau}\boldsymbol{Q}\boldsymbol{\tau} + \beta\right)\right\} \end{aligned}$$

for a tri-diagonal matrix $\boldsymbol{Q}$ with diagonal elements equal to two except first and last element which are one, and the off-diagonal elements equal to -1. We recognize the posterior as the core of an inverse gamma IG(α^*, β^*) with shape $\alpha^* = \alpha + \frac{1}{2}(T-1)$ and scale $\beta^* = \beta + \frac{1}{2}\boldsymbol{\tau}\boldsymbol{Q}\boldsymbol{\tau}$.

d) Acceptance probability

Let $\mathcal{I} \subseteq \{1, 2, ..., 366\}$ be a set of time indices, and let $-\mathcal{I} = \{1, 2, ..., 366\} \setminus \mathcal{I}$. Furthermore, let $\boldsymbol{\tau}'$ denote the proposed values for $\boldsymbol{\tau}$. The MH step needs an acceptance probability denoted α for the proposed values $\boldsymbol{\tau}'_{\mathcal{I}}$. By using iterative conditioning we can write the acceptance probability as

$$\alpha(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \min\left(1, \frac{P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})}{P(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})} \frac{Q(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})}{Q(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})}\right),$$

where our prior proposal distribution is $Q(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)$. By considering

$$\begin{aligned}
P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) &= \frac{P(\boldsymbol{\tau}'_{\mathcal{I}}, \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})}{P(\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})} \\
&= \frac{P(\boldsymbol{y}|\boldsymbol{\tau}'_{\mathcal{I}}, \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2) P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2) P(\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2) P(\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)} \\
&= \frac{\overbrace{P(\boldsymbol{y}|\boldsymbol{\tau}'_{\mathcal{I}}, \boldsymbol{\tau}_{-\mathcal{I}})}^{\text{Conditionally independent}} \quad P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)} \\
&= \frac{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}'_{\mathcal{I}}) P(\boldsymbol{y}_{-\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}) P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}
\end{aligned}$$

and equally

$$P(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \frac{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}_{\mathcal{I}}) P(\boldsymbol{y}_{-\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}) P(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}$$

we can rewrite the acceptance probability as

$$\begin{aligned}
\alpha(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) &= \min\left(1, \frac{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}'_{\mathcal{I}}) P(\boldsymbol{y}_{-\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}) P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2) / P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}_{\mathcal{I}}) P(\boldsymbol{y}_{-\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}) P(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2) / P(\boldsymbol{y}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)} \frac{P(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{P(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}\right) \\
&= \min\left(1, \frac{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}'_{\mathcal{I}})}{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}_{\mathcal{I}})}\right),
\end{aligned}$$

which is the minimum of 1 and the ratio of likelihoods conditioned on the proposed values and the old values.

e) Individual implementation

In this section we implement an MCMC sampler for the posterior $P(\boldsymbol{\pi}, \sigma_u^2|\boldsymbol{y})$. For the conditional prior, $P(\tau_t|\boldsymbol{\tau}_{-t}, \sigma_u)$, we use MH steps sequentially, and for σ_u we use Gibbs steps. The MH steps acceptance probability for updating individual τ_t parameters can be rewritten to improve computation time. In section 1.d) we found that the acceptance probability is a ratio of likelihoods, which in the individual update case is given by

$$\begin{aligned}
\frac{P(y_t|\tau'_t)}{P(y_t|\tau_t)} &= \frac{\pi(\tau'_t)^{y_t}(1 - \pi(\tau'_t))^{n_t - y_t}}{\pi(\tau_t)^{y_t}(1 - \pi(\tau_t))^{n_t - y_t}} \\
&= \frac{\exp\{y_t \tau'_t - n_t \ln(1 + e^{\tau'})\}}{\exp\{y_t \tau_t - n_t \ln(1 + e^{\tau})\}} \\
&= \exp\left\{y_t(\tau'_t - \tau_t) + n_t \ln\left(\frac{1 + e^{\tau}}{1 + e^{\tau'}}\right)\right\}, \qquad (2)
\end{aligned}$$

for which we chose to use the logarithm seen in the function `logAccRatio`. Thus, for our random walk we accept the proposed value if $\ln(r)$ is smaller than the logarithm of the ratio of likelihoods shown in Equation

(2). The $\tau_{\mathcal{I}}$ parameters for day $\mathcal{I} = t$ and days $-\mathcal{I}$ are multivariate Gaussian,

$$\begin{pmatrix} \tau_t \\ \boldsymbol{\tau}_{-t} \end{pmatrix} \sim MVN\left(\begin{pmatrix} \mu_t \\ \boldsymbol{\mu}_{-t} \end{pmatrix}, \begin{pmatrix} Q_{t,t} & \mathbf{Q}_{t,-t} \\ \mathbf{Q}_{-t,t} & \mathbf{Q}_{-t,-t} \end{pmatrix}^{-1}\right).$$

The conditional mean and precision used to draw the proposed value for τ_t are given by

$$\mu_{t|-t} = \mu_t - Q_{t,t}^{-1}\boldsymbol{Q}_{t,-t}\left(\boldsymbol{\tau}_{-t} - \boldsymbol{\mu}_{-t}\right) = -Q_{t,t}^{-1}\boldsymbol{Q}_{t,-t}\boldsymbol{\tau}_{-t}$$
$$Q_{t|-t} = Q_{t,t}.$$

For the implementation we use the given values $\alpha = 2$ and $\beta = 0.05$ for the response given in Equation (1), and we set the initial value of σ_u^2 to be 0.05. Initial values for τ are drawn from a standard normal distribution. We run the MCMC sampler for a total of $N = 50000$ iterations. In the following chunk are the functions used for the individually updating sampler followed by the chunk for which we run the sampler.

```
link = function(tau){
  # Expit link
  return(exp(tau)/(1+exp(tau)))
}

logbin = function(n, y, tau){
  # Remake and use this
  return(y*log(1+exp(-tau)) - (n-y)*log(1+exp(tau)))
}

logAccRatio = function(n, y, tauProp, tau) {
  return(y*(tauProp - tau) + n* log((1+exp(tau))/(1+exp(tauProp))))
}

mhFirst = function(tau, sigma, yt, t, norm_it, unif){
  # Function to take the first MH step, i.e., for t=1
  mu_ab = tau[2]
  prop_tau = norm_it*sigma + mu_ab
  n = 39
  # ratio = acceptRatio(n, yt, prop_tau, tau[t])
  ratio = logAccRatio(39, yt, prop_tau, tau[t])
  # if (runif(1) < min(c(1,ratio))){
  if (unif < min(c(1,ratio))){
    return(list(tau=prop_tau, accepted=1))
  }
  else{return(list(tau=tau[t], accepted=0))}
}

mhLast = function(tau, sigma, yt, t, norm_it, unif){
  # Function to take the last MH step, i.e., for t=366
  mu_ab = tau[365]
  prop_tau = norm_it*sigma + mu_ab
  # ratio = acceptRatio(n, yt, prop_tau, tau[t])
  ratio = logAccRatio(39, yt, prop_tau, tau[t])
  if (unif < min(c(1,ratio))){
    return(list(tau=prop_tau, accepted=1))
  }
  else{return(list(tau=tau[t], accepted=0))}
}
```

```
mcmcIndivid = function(N, dt, sigma0=0.05){
  # Allocate memory
  Ttot = 366
  tau = matrix(NA, nrow=N, ncol = Ttot)
  sigma = numeric(length = N)
  tau_i = numeric(length = Ttot)
  normMat = matrix(rep(rnorm(Ttot), N), nrow = N, ncol=Ttot)
  # unifMat = matrix(rep(runif(Ttot), N), nrow=N, ncol=Ttot)
  unifMat = matrix(rep(log(runif(Ttot)), N), nrow=N, ncol=Ttot) # log uniform

  # Find init vals
  tau[1,] = rnorm(Ttot) # init tau drawn from normal distr.
  sigma[1] = sigma0

  # Run mcmc for N iterations
  accepted = 0
  for (i in 2:N){
    tau_i = tau[i-1,]
    sigma_i = sqrt(sigma[i-1])

    # Take first MH step for t=1
    mhStep = mhFirst(tau_i, sigma_i, n.rain[1], 1, normMat[i,1], unifMat[i,1])
    tau_i[1] = mhStep$tau
    accepted = accepted + mhStep$accepted

    # Perform MH steps for 1<t<366
    for (t in 2:(Ttot-1)){
      mu_ab    = 1/2 * (tau_i[t-1] + tau_i[t+1])
      prop_tau = normMat[i,t]*sigma_i/2 + mu_ab
      ratio    = logAccRatio(n.years[t], n.rain[t], prop_tau, tau_i[t])
      if (unifMat[i,t] < min(c(1,ratio))){
        tau_i[t] = prop_tau
        accepted = accepted + 1
      }
      # else{tau[i,t] = tau_i[t]}
    }

    # Take last MH step for t=366
    mhStep = mhLast(tau_i, sigma_i, dt$n.rain[366], 1, normMat[i,366], unifMat[i,366])
    tau_i[366] = mhStep$tau
    tau[i,] = tau_i
    accepted = accepted + mhStep$accepted

    # Squared diff. of tau vec.
    # tQt = sum((tau[i,-Ttot] - tau[i,-1])^2) # this sim tau vals.
    tQt = sum(diff(tau[i,])^2) # this sim tau vals.

    # Gibbs step (Draw from IG)
    sigma[i] = 1/rgamma(1, shape=2 + (Ttot-1)/2, scale=0.05 + 0.5*tQt) # Gibbs inline

    setTxtProgressBar(pb,i)
  }
  close(pb)
```

```
  return(list(tau=tau, sigma=sigma, accProb = accepted/(N*Ttot)))
}
```

```
a          = 0
Q          = triDiag(diagonal = 2, upper = -1, lower = -1, nrow = 366)
Q[1,1]     = 1
Q[366,366] = 1

set.seed(321)
N       = 50000
pb      = txtProgressBar(min = 0, max = N, initial = 0)
ptm     = proc.time()
results = mcmcIndivid(N, rain)
```

```
## ================================================================================
```

```
time    = proc.time() - ptm
time
```

```
##    user  system elapsed
##   47.28    0.12   47.55
```

In the printout we see that computation time was 47.55. The following chunk contain functions that compute some sought values and make desired figures.

```
CImean = function(p){
  # Computes mean and upper,lower quantiles of vector.
  c(mean=mean(p), quantile(p, probs = c(0.025,0.975)))
}

plotTAH = function(p, hcol = "cyan3", xlab = "", ylab = "", burn=3000){
  # Plots trace, autocorr and hist of probs
  l = quantile(p, probs = 0.025)
  u = quantile(p, probs = 0.975)
  plot(p, type="l", xlab = "Iterations", ylab="Probability")
  abline(h=mean(p), col="red")
  abline(v=burn, lty=2, col="gray")
  acf(p, main="")
  hist(p, nclass=200, prob=T, main="", xlab="Probability", xlim=c(l-0.01,u+0.01))
  abline(v = l, col=hcol)
  abline(v = u, col=hcol)
}
pSeq = apply(results$tau, 2, link)
```

```
par(mfrow=c(3,3))
plotTAH(pSeq[,1])
plotTAH(pSeq[,201])
plotTAH(pSeq[,366])
```

The vertical red line in the trace plots to the left in Figure 2 show the mean of $\pi(\tau_t)$ over all 50000 iterations, for $t = 1, 201, 366$. The horizontal blue line is at the 3000^{nd} iteration. The vertical blue lines in the histogram

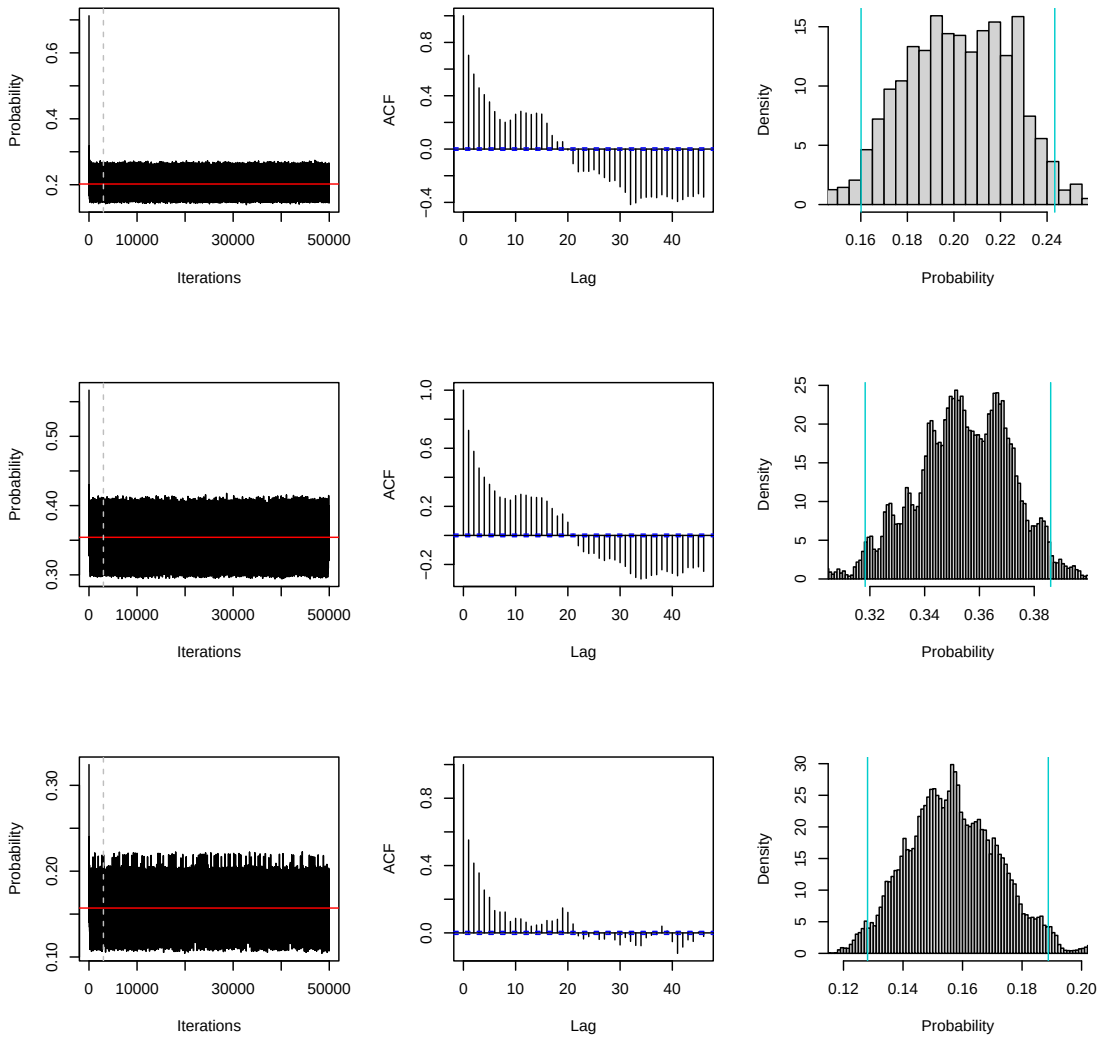


Figure 2: Traceplot, autocorrelation and histogram plots for $\pi(\tau_1), \pi(\tau_{201})$ and $\pi(\tau_{366})$ from top to bottom.

Table 1: mean and stuff

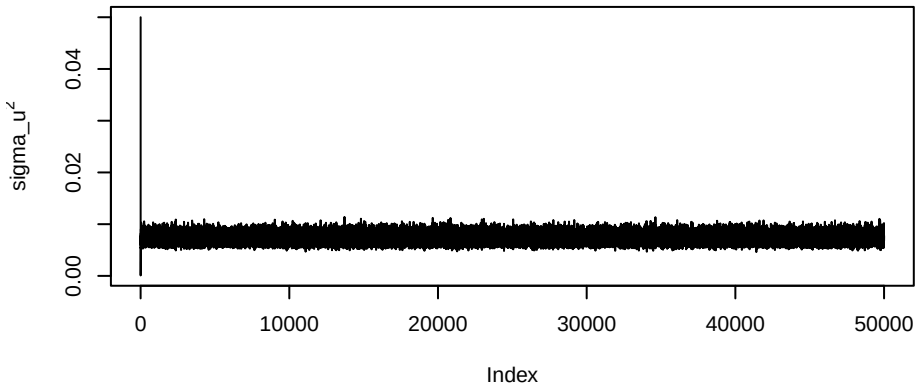
	mean	2.5%	97.5%
1	0.2022	0.1601	0.2433
1 B	0.2022	0.1601	0.2432
201	0.3542	0.3183	0.3860
201 B	0.3542	0.3183	0.3859
366	0.1570	0.1281	0.1888
366 B	0.1570	0.1281	0.1886
sigma	0.0074	0.0057	0.0093
sigma B	0.0074	0.0057	0.0093

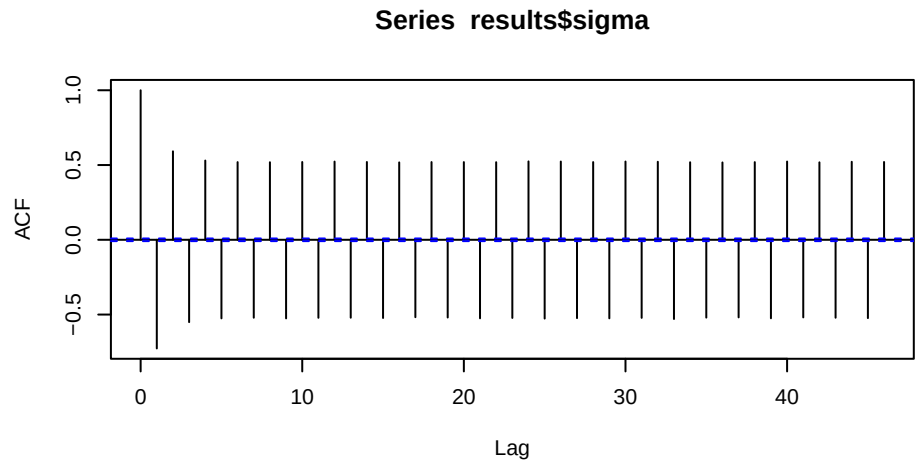
plot show the 95% credible intervals for $\pi(\tau_t)$. The traceplot bares resemblance to that of a random walk for all τ_t after approximately the 3000^{nd} iteration. We notice a decrease in autocorrelation and the histogram show that $\pi(\tau_t)$ seem normally distributed.

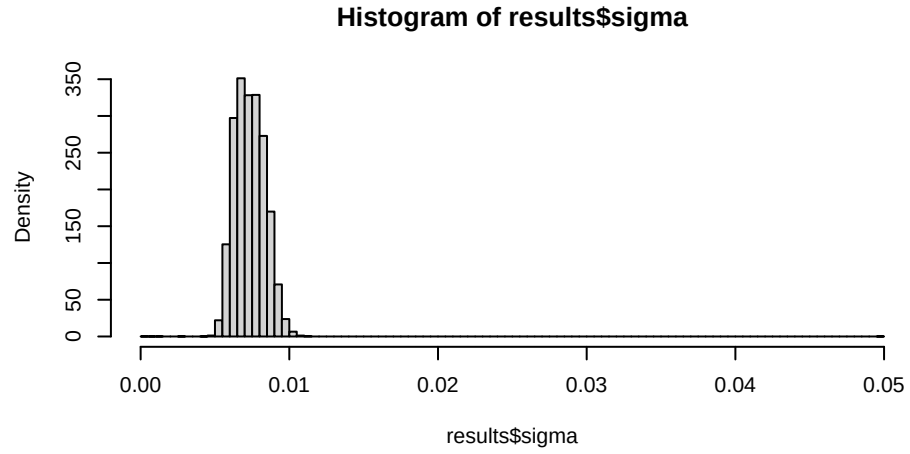
```
pSeqCI = apply(pSeq, 2, CImean) # Compute mean and credibility interval

# Burn
burn           = 3000
resultsB       = results
resultsB$tau   = results$tau[burn:N,]
resultsB$sigma = results$sigma[burn:N]
pSeqB          = pSeq[burn:N,]
pSeqCIB        = apply(pSeqB, 2, CImean) # Compute mean and CI
ciAndMean      = rbind((pSeqCI[,1]), (pSeqCIB[,1]),
                   (pSeqCI[,201]), (pSeqCIB[,201]),
                   (pSeqCI[,366]), (pSeqCIB[,366]),
                   CImean(results$sigma), CImean(resultsB$sigma))
rownames(ciAndMean) = c("1","1 B", "201","201 B", "366","366 B", "sigma", "sigma B")

knitr::kable(round(ciAndMean,4), caption = "mean and stuff")
```

Burning some of the early iterations, here the first 3000, has some impact, but nothing major due to the fast convergence of MCMC sampler, as we can see in Table 1.

```
par(mfrow=c(3,1))
yl = expression(sigma_u^2)
plot(results$sigma, type="l", ylab=yl)
acf(results$sigma)
hist(results$sigma, nclass=100, prob=T)
```








```
ylim = c(min(rain$n.rain/rain$n.years), max=max(pSeq[1,]))
par(mfrow=c(1,1))
plot(rain$day,pSeq[1,], type = "l",
     ylab=expression(pi), xlab="Day",
```

```
     ylim = ylim, col="gray")
lines(rain$day,pSeq[N/2,], type = "l",
     ylim = ylim, col="cyan3")
lines(rain$day,pSeq[N,],
     ylim = ylim, col = "blue", lty=1)
legend(x="topright", legend = c("Initial values", "Halfway", "Last iteration"), lty = c(1,1,1),
       col=c("gray", "cyan3", "blue"), bg="transparent")
```

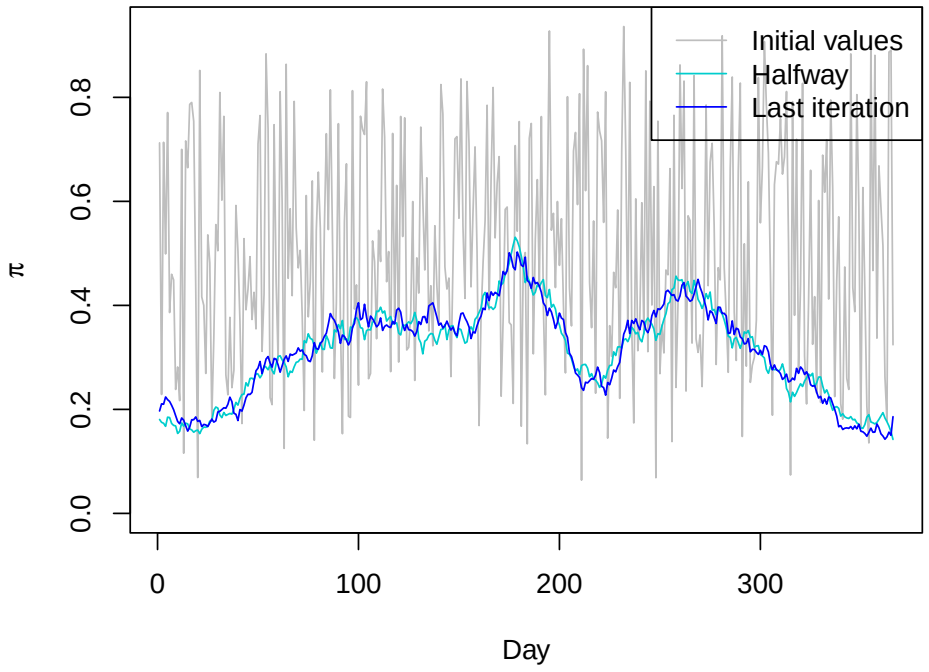


Figure 3: Make cap

The initial values of $\boldsymbol{\pi}$ show no resemblance to the Tokyo rainfall dataset probabilities in Figure 3

```
gg1e.df = data.frame(probPreds = pSeq[N,], l = pSeqCI[2, ], u = pSeqCI[3,],
                     meanPreds = pSeqCI[1,], probsReal = n.rain/n.years,
                     day = day)
ggplot(data = gg1e.df, aes(x = day, y = probPreds)) +
  # geom_ribbon(mapping = aes(ymin = l,ymax = u, fill = "CI"), alpha = 0.2) +
  geom_line(aes(color = "tau", linetype="tau")) +
  geom_line(aes(y = meanPreds, color = "mean", linetype="mean")) +
  geom_point(aes(y = probsReal, color = "n.rain/n.years",
                 size="n.rain/n.years"), alpha = 0.3) +
  scale_linetype_manual(
    name=" ",
    breaks = c("tau", "mean", "n.rain/n.years"),
    values = c(2, 1, NA),
    labels=expression(pi*(tau),'Mean','Data probability')
```

```
  # breaks = c("tau", "mean"),
  # values = c(2, 1),
  # labels=expression(pi*(tau),'Mean')
  # values = c("tau"=1, "mean"=2, "n.rain/n.years"=2)
) +
scale_size_manual(
  name=" ",
  breaks = c("tau", "mean", "n.rain/n.years"),
  values = c(NA, NA, 0.1),
  labels=expression(pi*(tau),'Mean','Data probability')
  ) +
scale_color_manual(
  name=" ",
  # breaks = c("tau", "mean", "n.rain/n.years"),
  values=c("tau"="green", "mean"="black","n.rain/n.years"="blue"),
  # values=c("black", "green","pink"),
  labels=expression(pi*(tau),'Mean','Data probability')
) +
labs(y="Probabilities", x="Day") +
theme_minimal()
```

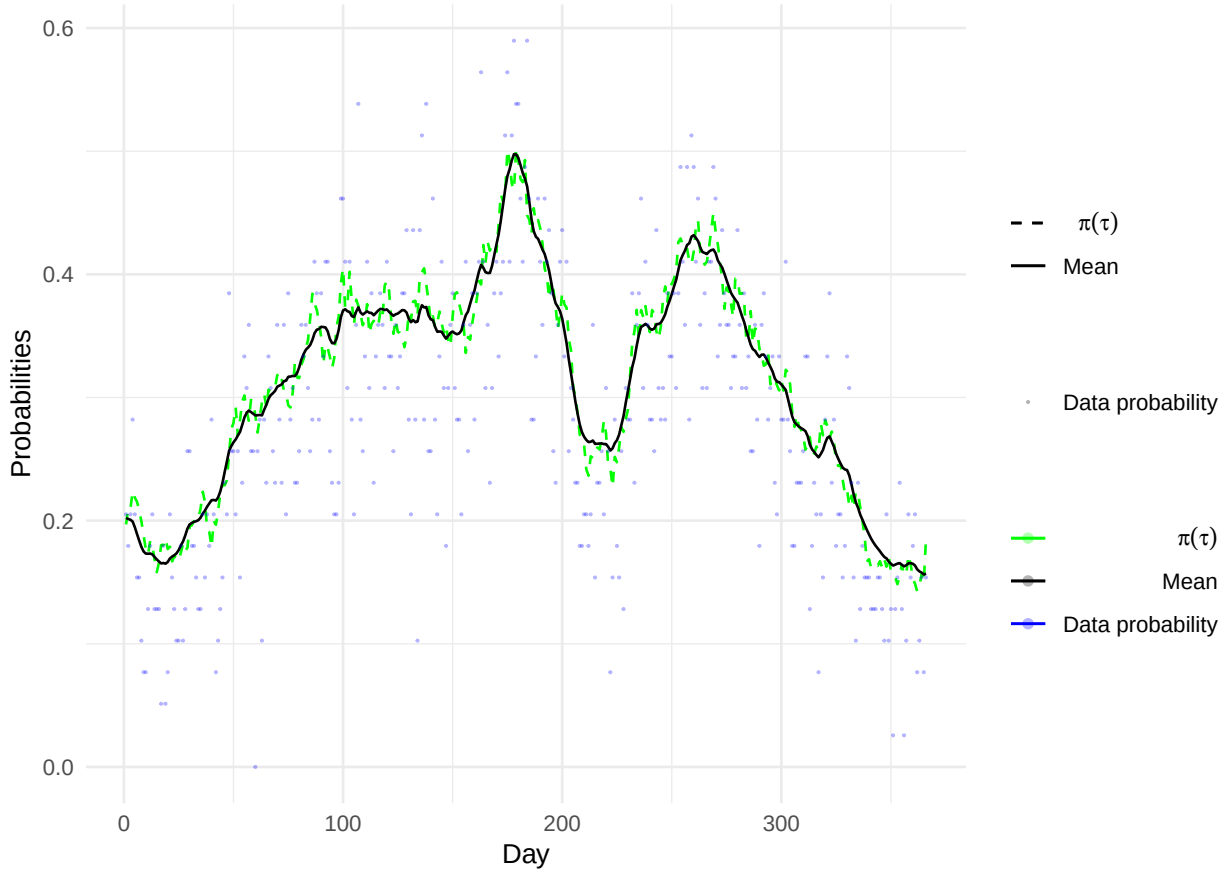


Figure 4: The black line is the mean of all predicted probabilities π excluding the burned predictions. The dashed green and blue lines are last iterations MCMC samples and the data probabilities, respectively.

f) Blocking implementation

Now we will go from sequentially updating τ_t to perform block updates for each iteration. We will start by showing what mathematical implications that comes from blocking before we show the implementation and discuss the results. Rather than updating individual τ_t parameters we will use a conditional prior proposal involving $P(\boldsymbol{\tau}_{(a,b)}|\boldsymbol{\tau}_{-(a,b)})$, where $\boldsymbol{\tau}_{(a,b)} = (\tau_a, ..., \tau_b)$ for interval length $M = b - a$. For intervals of length $M > 1$ we can simplify the acceptance probability as we did in section , only now

$$\frac{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}'_{\mathcal{I}})}{P(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}_{\mathcal{I}})} = \exp\left\{\sum_{\mathcal{I}} y_i(\tau'_i - \tau_i) + n_i \ln\left(\frac{1+e^{\tau_i}}{1+e^{\tau'_i}}\right)\right\}, \tag{3}$$

where $\mathcal{I} = (a, b)$. The conditional mean and precision used to draw the proposed values $\boldsymbol{\tau}_{\mathcal{I}}$ are given by

$$\boldsymbol{\mu}_{\mathcal{I}|-\mathcal{I}} = \boldsymbol{\mu}_{\mathcal{I}} - \mathbf{Q}^{-1}_{\mathcal{I},\mathcal{I}}\mathbf{Q}_{\mathcal{I},-\mathcal{I}}\left(\boldsymbol{\tau}_{-\mathcal{I}} - \boldsymbol{\mu}_{-\mathcal{I}}\right) = -\mathbf{Q}^{-1}_{\mathcal{I},\mathcal{I}}\mathbf{Q}_{\mathcal{I},-\mathcal{I}}\boldsymbol{\tau}_{-\mathcal{I}}$$
$$\mathbf{Q}_{\mathcal{I}|-\mathcal{I}} = \mathbf{Q}_{\mathcal{I},\mathcal{I}}.$$

Assuming $1 < M < T - 2$, we get three different types of precision matrices which can be precomputated. Namely, one when the first day is included, the second where neither the first nor the last day is included, and the third where the last day is included in the set of days ⟩. There are also only three different precision matrices for one iteration for the same day sets mentioned which also can be precomputated for each iteration. precomputed or precomputated?. The shape and size of said deflectors of the $\mathbf{Q}$ matrix will not be discussed further, but can be computed from the `Qprecomp` function seen in the chunk below. The chunk also include the function for computing the acceptance ratio from Equation (3) and the implementation of the MCMC sampler.

```
acceptRatioBlock = function(I, tauProp, tauPrev){
  return(exp(sum(n.rain[I]*(tauProp - tauPrev)) +
              sum(rain$n.years[I]*(log(1+ exp(tauPrev)) -
                                     log(1+exp(tauProp)))
                  )
              )
         )
}

Qprecomp = function(M, Ttot=366){
  ## Make intervals I
  n.sets = ceiling(Ttot/M)
  I = matrix(1:(M*(n.sets-1)), ncol = n.sets-1, byrow=F)
  Ires = (I[M,n.sets-1]+1):Ttot

  # Make Q
  Q = triDiag(diagonal = 2, upper = -1, lower = -1, nrow = Ttot)
  Q[1,1] = 1
  Q[Ttot,Ttot] = 1
  # Find the three Q_AA
  Qaa1 = Q[1:M,1:M]
  Qaa2 = Q[2:(M+1), 2:(M+1)]
  Qaa3 = Q[Ires, Ires]
  Qaa1inv = solve(Qaa1)
  Qaa2inv = solve(Qaa2)
  Qaa3inv = solve(Qaa3)

  Qmult = list(Qmult1 = -Qaa1inv %*% Q[1:M, -(1:M)])
```

```
  for (i in 2:(n.sets-1)){
    Qmult = append(Qmult, list(-Qaa2inv %*% Q[I[,i], -I[,i]]))
  }
  Qmult = append(Qmult, list(-Qaa3inv %*% Q[Ires, -Ires]))
  names(Qmult) = sprintf("Qmult%d", 1:(n.sets))

  return(list(Qaa1inv = Qaa1inv, Qaa2inv = Qaa2inv, Qaa3inv = Qaa3inv,
              Qmult = Qmult,
              n.sets = n.sets, I = I, Ires = Ires))
}

mcmcBlock = function(N, M=10, sigma0=0.1, tau0=rnorm(366)){
  # Allocate memory
  tau      = matrix(NA, nrow = N, ncol = Ttot) # (N x T)
  sigma    = numeric(length = N)
  # Assign initial values
  tau[1,]  = tau0
  sigma[1] = sigma0
  # Assign Q matrices for easier access
  Qm       = Q$Qmult # precomputed -QaaInverse * Qab
  n.sets   = Q$n.sets
  Imat     = Q$I # (M x (n.sets-1))
  Ires     = Q$Ires

  # Simulate N times
  accepted = 0
  for (i in 2:N){
    # Perform first MH block step using upper left sub matrices of Q
    mhBlock = mhBlockStep(tau[i-1,], Imat[,1], sigma[i-1]*Q$Qaa1inv, Qm$Qmult1)
    tau[i, Imat[,1]] = mhBlock$tau
    accepted = accepted + mhBlock$accepted

    # Precompute sigma_u^2 * Q_AA^-1 for all blocks where 1<t<366
    Sigma.mid = sigma[i-1]*Q$Qaa2inv
    # Perform MH block steps for days 1<t<366
    for (t in 2:(n.sets-1)){
      mhBlock = mhBlockStep(tau[i-1,], Imat[,t], Sigma.mid, Qm[[t]])
      tau[i, Imat[,t]] = mhBlock$tau
      accepted = accepted + mhBlock$accepted
    }

    # Perform last MH block step using lower left sub matrices of Q
    mhBlock.last = mhBlockStep(tau[i-1,], Ires, sigma[i-1]*Q$Qaa3inv, Qm[[n.sets]])
    tau[i, Ires] = mhBlock.last$tau
    accepted = accepted + mhBlock.last$accepted

    # Perform Gibbs step for sigma_u^2
    tQt = sum((diff(tau[i,]))^2)
    sigma[i] = 1/rgamma(1, shape=2 + (Ttot-1)/2, scale=0.05 + 0.5*tQt)

    setTxtProgressBar(pb,i)
  }
  close(pb)
```

```
  results = list(tau, sigma, accepted/(N*Ttot))
  names(results) = c(paste0('tau',M), paste0('sigma',M), paste0('accProb',M))
  return(results)
}

mhBlockStep = function(tauPrev, I, Sigma, Qm.I){
  # Inputs
  # tauPrev : Previous sim tau values
  # I       : Indices for this block
  # Sigma   : sigma_u^2 * Q_AA^-1
  # Qm.I    : -Q_AA^-1 x Q_AB

  mu.ab = Qm.I %*% tauPrev[-I]
  tauProp = mvrnorm(n=1, mu=mu.ab, Sigma=Sigma)
  ratio = acceptRatioBlock(I, tauProp, tauPrev[I])
  # Random walk 1
  if (runif(1) < min(c(1,ratio))){
    return(list(tau=tauProp, accepted=length(I)))
  }
  else{return(list(tau=tauPrev[I], accepted=0))}
}

M = 10
N = 5000
Ttot = 366
```

In the following chunks we will run the block updating MCMC sampler for $N = 5000$ simulations with intervals of length $M = 10$. First we present the computation time followed by some visualizations for an overview of the results. Then we will discuss burning and do some computations and visualizations on the burned results.

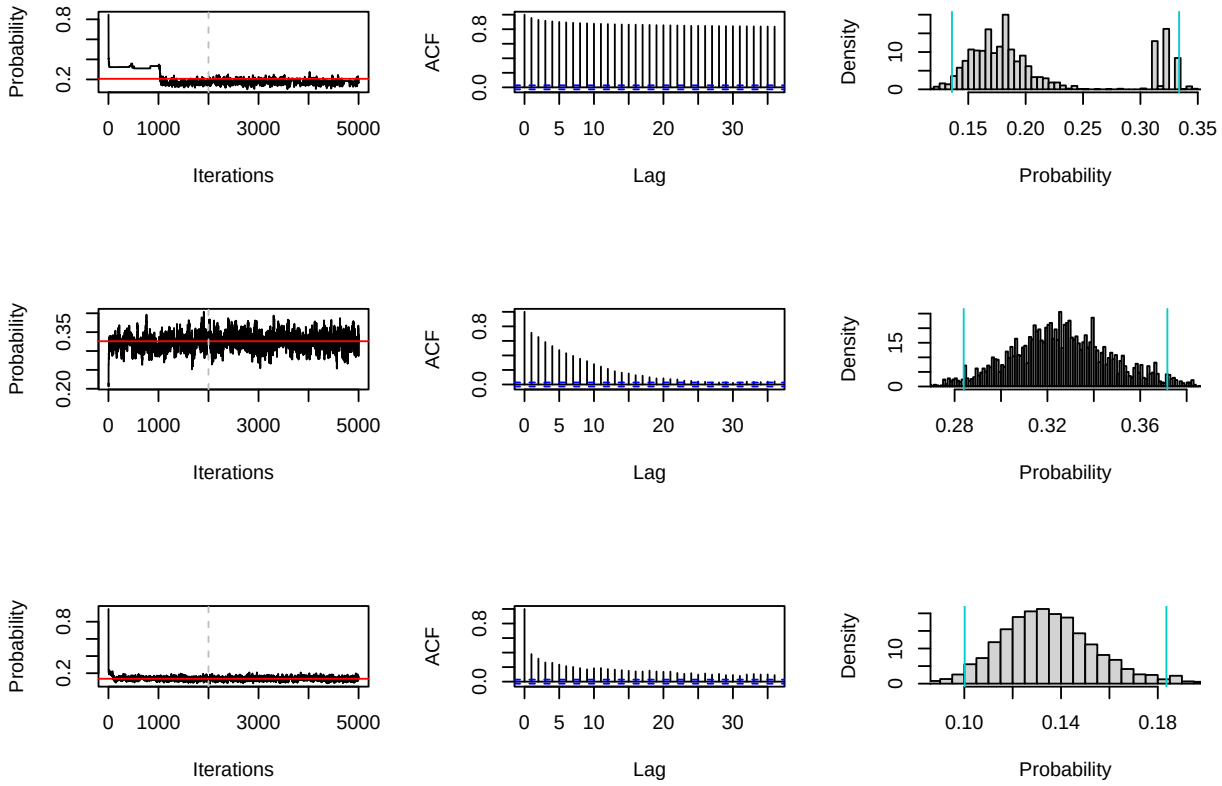
```
Q = Qprecomp(M)
pb = txtProgressBar(min = 0, max = N, initial = 0)
set.seed(321)
ptm = proc.time()
# profvis({results = mcmcBlock(N, rain, M)}) # For profiling
resultsBlock = mcmcBlock(N, M)
```

```
## ================================================================================
```

```
timeBlock = proc.time() - ptm
timeBlock
```

```
##    user  system elapsed
##   17.98    0.00   18.04
```

```
pSeq = apply(resultsBlock$tau, 2, link)
par(mfrow=c(3,3))
plotTAH(pSeq[,1], burn=2000)
plotTAH(pSeq[,201], burn=2000)
plotTAH(pSeq[,366], burn=2000)
```

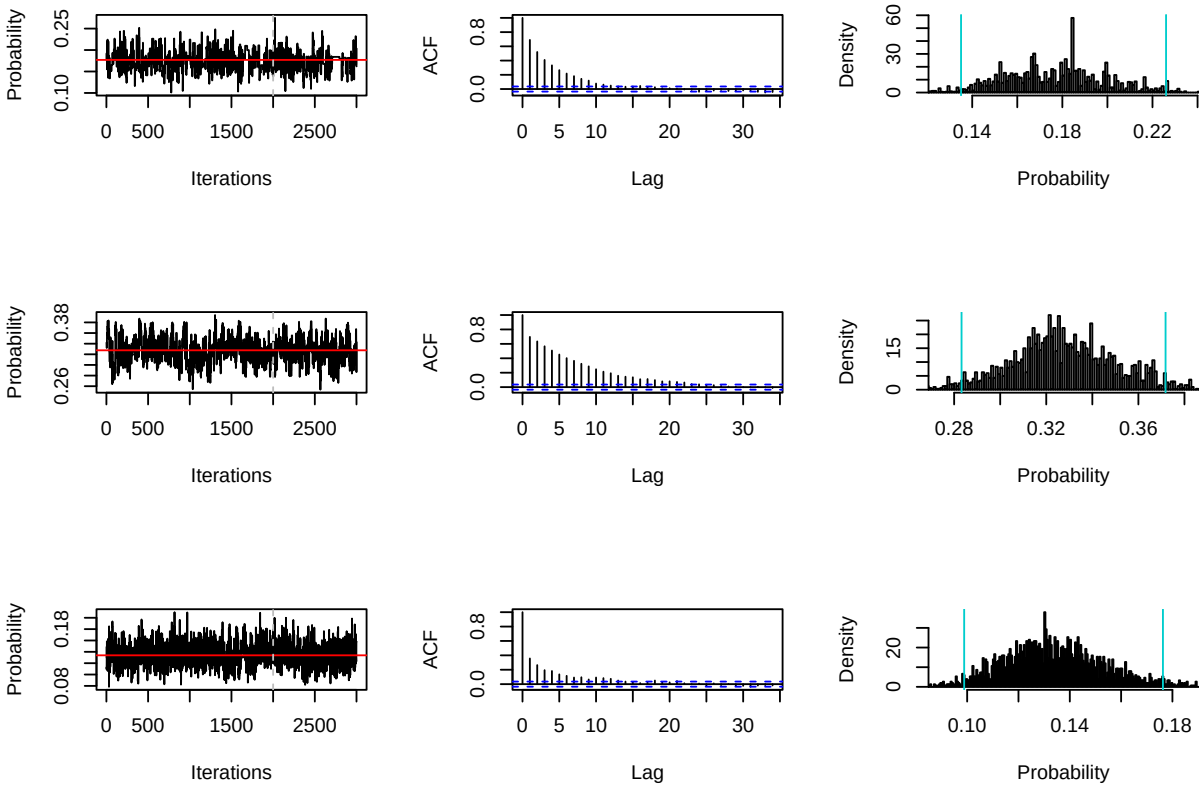



```
pSeqCI = apply(pSeq, 2, CImean) # Compute mean and credibility interval

# Burn
burn=2000
resultsB = resultsBlock
resultsB$tau = resultsBlock$tau[burn:N,]
resultsB$sigma = resultsBlock$sigma[burn:N]
pSeqB = pSeq[burn:N,]
pSeqCIB = apply(pSeqB, 2, CImean) # Compute mean and credibility interval
ciAndMean = rbind((pSeqCI[,1]), (pSeqCIB[,1]),
                  (pSeqCI[,201]), (pSeqCIB[,201]),
                  (pSeqCI[,366]), (pSeqCIB[,366]),
                  CImean(resultsBlock$sigma), CImean(resultsB$sigma))
rownames(ciAndMean) = c("1","1 B", "201","201 B", "366","366 B", "sigma", "sigma B")
# knitr::kable(round(ciAndMean,4), caption = "mean and stuff")
knitr::kable(round(ciAndMean,4), caption = "mean and stuff")
```

```
par(mfrow=c(3,3))
plotTAH(pSeqB[,1], burn=2000)
plotTAH(pSeqB[,201], burn=2000)
plotTAH(pSeqB[,366], burn=2000)
```

Table 2: mean and stuff

	mean	2.5%	97.5%
1	0.2061	0.1358	0.3337
1 B	0.1770	0.1350	0.2260
201	0.3259	0.2839	0.3717
201 B	0.3276	0.2832	0.3718
366	0.1354	0.1002	0.1836
366 B	0.1339	0.0988	0.1763
sigma	0.0044	0.0021	0.0061
sigma B	0.0051	0.0041	0.0062




```
gg1e.df = data.frame(probPreds = pSeq[N,], l = pSeqCI[2, ], u = pSeqCI[3,],
                     meanPreds = pSeqCI[1,], probsReal = n.rain/n.years,
                     day = day)
ggplot(data = gg1e.df, aes(x = day, y = probPreds)) +
  geom_line(aes(color = "tau", linetype="tau")) +
  geom_line(aes(y = meanPreds, color = "mean", linetype="mean")) +
  geom_point(aes(y = probsReal, color = "n.rain/n.years",
                 size="n.rain/n.years"), alpha = 0.3) +
  scale_linetype_manual(
    name=" ",
    breaks = c("tau", "mean", "n.rain/n.years"),
    values = c(2, 1, NA),
```

```
    labels = expression(pi*(tau),'Mean','Data probability')
  ) +
  scale_size_manual(
    name=" ",
    breaks = c("tau", "mean", "n.rain/n.years"),
    values = c(NA, NA, 0.1),
    labels = expression(pi*(tau),'Mean','Data probability')
    ) +
  scale_color_manual(
    name=" ",
    values = c("tau"="green", "mean"="black","n.rain/n.years"="blue"),
    labels = expression(pi*(tau),'Mean','Data probability')
  ) +
  labs(y="Probabilities", x="Day") +
  theme_minimal()
```

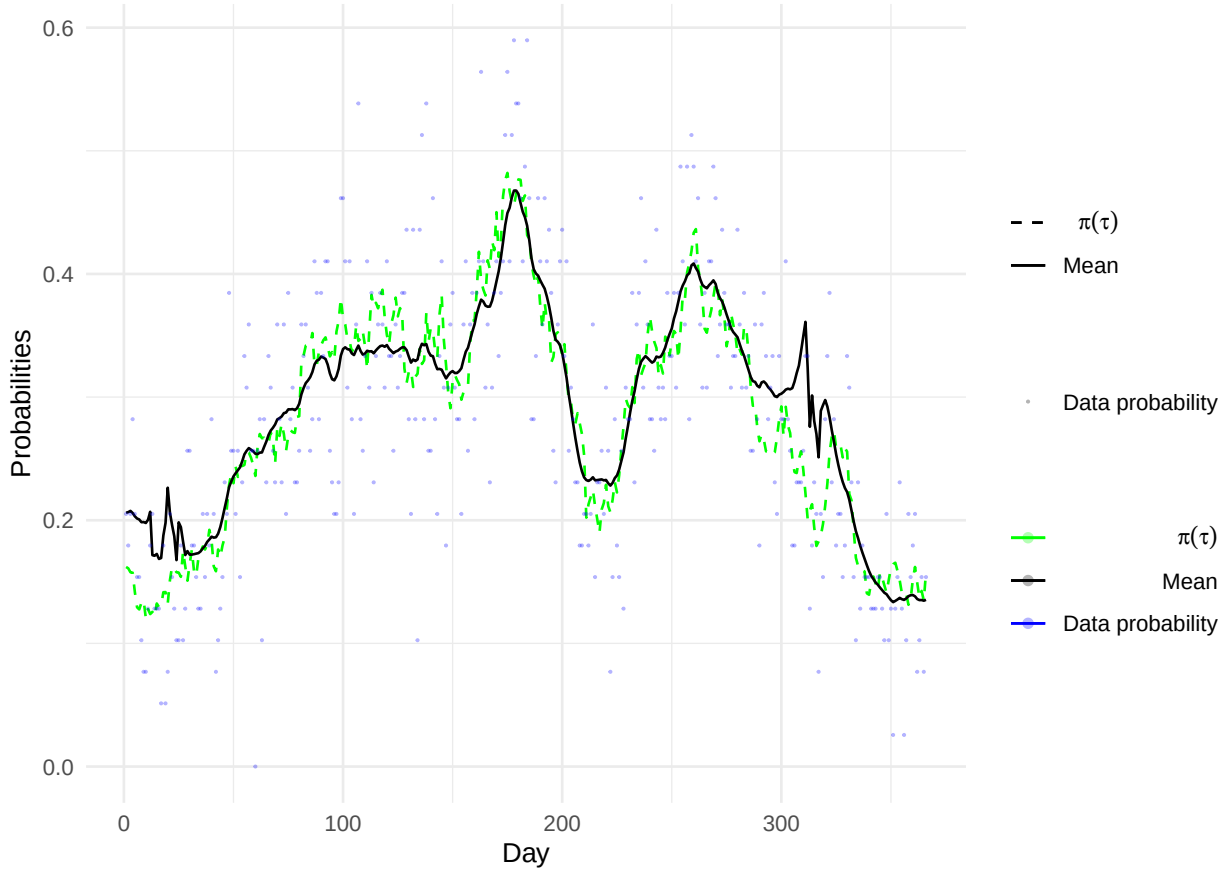


Figure 5: makeCap

```
N = 5000

Mlist = c(2, 5, 10, 15, 20, 25, 40, 80)
n.runs = length(Mlist)
resultList = list()
time = c()
```

```
set.seed(321)
tau0 = rnorm(366)

for (i in 1:length(Mlist)) {
    set.seed(321)
    pb = txtProgressBar(min = 0, max = N, initial = 0)
    Q = Qprecomp(Mlist[i], Ttot)
    start = proc.time()[3]
    resultList = append(resultList, mcmcBlock(N, Mlist[i], tau0 = tau0))
    time = c(time, proc.time()[3] - start)
}
```

```
## ================================================================================
## ================================================================================
## ================================================================================
## ================================================================================
## ================================================================================
## ================================================================================
## ================================================================================
## ================================================================================
```

The chunk above ran the MCMC sampler for $N = 5000$ with interval lengths $\boldsymbol{M} = \{2, 5, 10, 15, 20, 25, 40, 80\}$. In Figure 6 produced by the chunk below we see that the acceptance probability decrease with increasing interval lengths M. For a RW(1) model, a rule of thumb is to have acceptance probability $\alpha \in (0.2, 0.5)$ which coincide with the acceptance probability for $M = 20$. We also notice that the computation time decrease for interval lengths $M \leq 20$ before it starts to increase. Thus, based on these results, $M = 20$ seems like a good candidate for interval length.

```
accProbsBlock = c()
for (m in Mlist) {
    accProbsBlock = c(accProbsBlock, resultList[[paste0("accProb", m)]])
}
par(mfrow = c(1, 1))
plot(Mlist, time, ylim = c(0, max(c(time, 100 * accProbsBlock))), pch = 16, xaxt = "n",
    xlab = "M", ylab = "Time/acceptance probability (%)")
points(Mlist, accProbsBlock * 100, col = "cyan3", pch = 16)
abline(h = c(20, 50), lty = 2)
axis(1, at = Mlist)
```

To further investigate candidates for interval lengths we consider the traceplots in Figure 7 produced by the chunk below. Since all fits were run for only \$N= 5000 \$ iterations, it is difficult to say whether the chain has fully converged. The three plot rows in the bottom show clear signs of not convergence. Still, the resulting traceplots does not rule out our favorable candidate from before, namely $M = 20$, showing promising convergence rate.

```
dToPlot = c(1, 201, 366)
par(mfrow = c(length(Mlist), length(dToPlot)), mar = c(2, 2, 0.1, 1))
for (i in 1:length(Mlist)) {
    xaxt = ifelse(i == length(Mlist), "s", "n")
    for (t in dToPlot) {
        plot(resultList[[paste0("tau", Mlist[i])]][, t], type = "l", xlab = "", ylab = "",
            xaxt = xaxt)
```

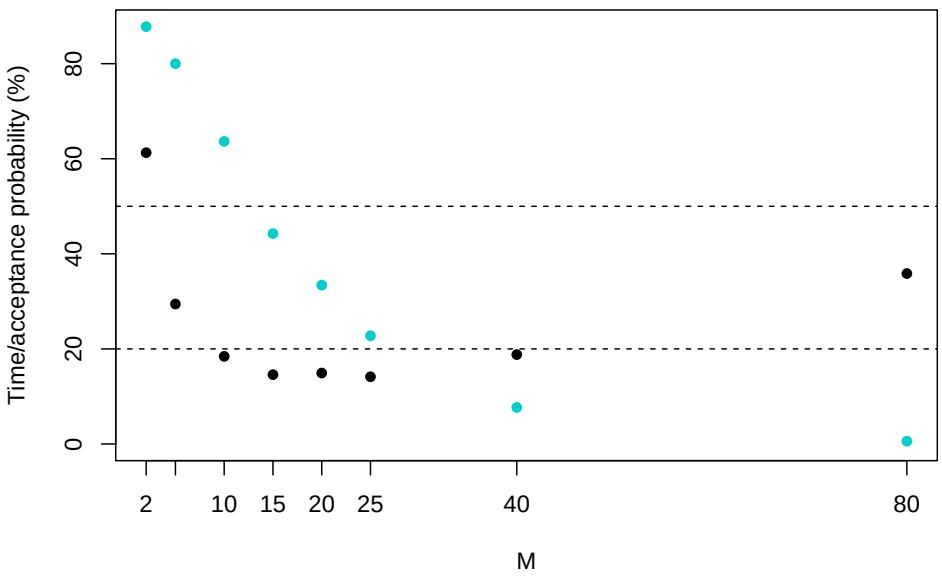


Figure 6: Computation time for different values of M along with the respective acceptance probability. Dashed horizontal lines show desired acceptance probability for RW(1).

```
    }
}
```

So, some values of M show promicing results, but we still need to compare them to the real data. Since all traceplots in Figure 7 show signs of not convergence for early iterations we chose to burn some values. The amount burned is shown in the chunk below

```
burnTune = 1000  # burn amount for M tuning
pMeanBlockMat = matrix(NA, nrow = length(Mlist), ncol = Ttot)
for (i in 1:length(Mlist)) {
    pMeanBlockMat[i, ] = apply(link(resultList[[paste0("tau", Mlist[i])]][burnTune:N,
        ]), 2, mean)
}
a = 0
```

```
colfunc <- colorRampPalette(c("red", "yellow", "springgreen", "royalblue"))
c = colfunc(length(pMeanBlockMat[, 1]))
diffTune = c()
pReal = n.rain/n.years
# plot(day,pReal, pch=4, cex=0.5)
par(mfrow = c(ceiling(length(Mlist)/2), 2), mar = c(1, 2, 1, 1))
for (i in 1:length(pMeanBlockMat[, 1])) {
    ifelse(i < length(pMeanBlockMat[, 1]) - 1, NA, par(mar = c(2, 2, 1, 1)))
    yaxt = ifelse(i%%2, "s", "n")
    xaxt = ifelse(i < length(pMeanBlockMat[, 1]) - 1, "n", "s")
```

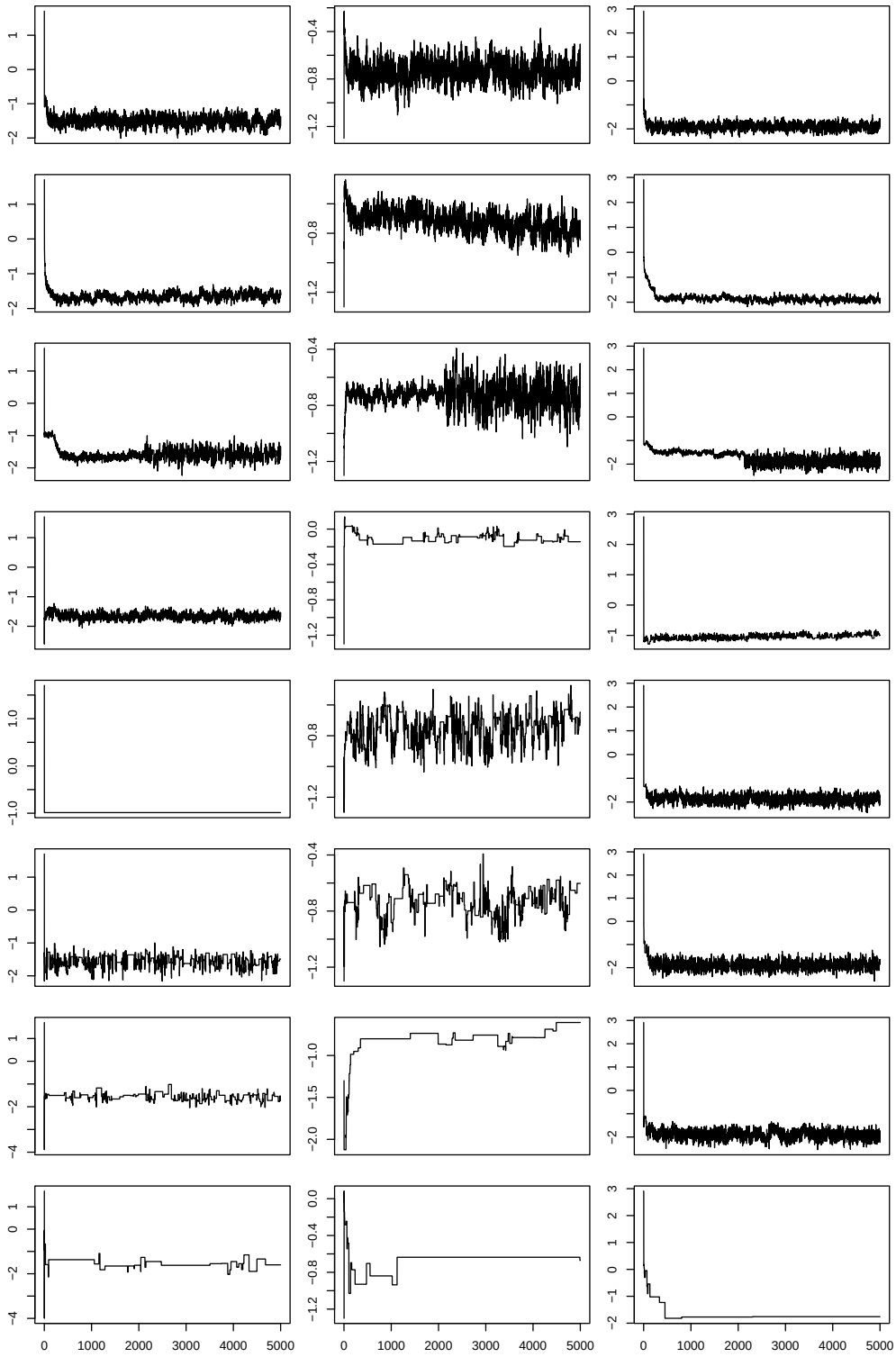
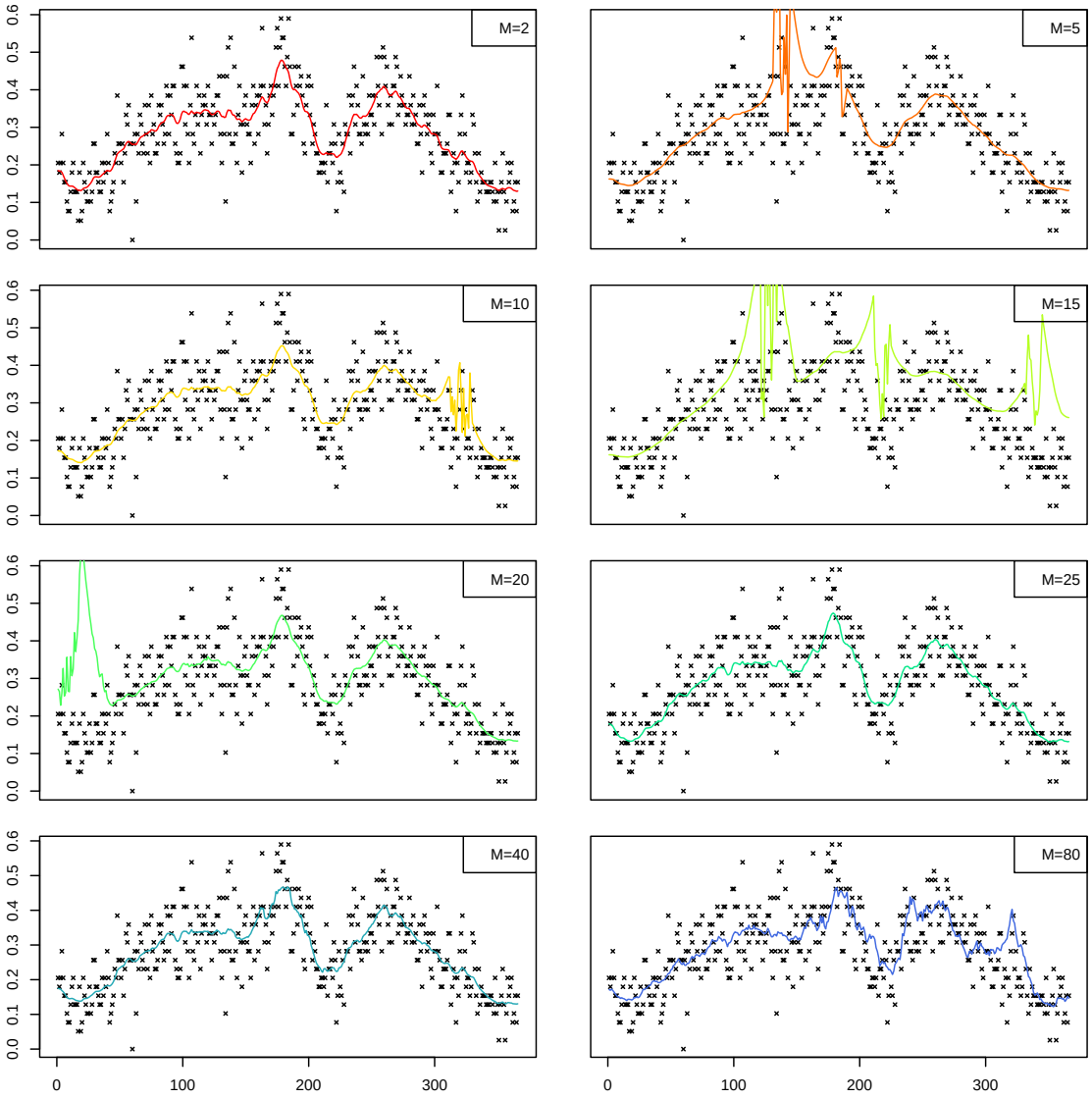


Figure 7: Trace plots of $\boldsymbol{\pi}$ for low to high value of the tuning param M from top to bottom, respectively, for different days t.

```
    xl = paste0("M = ", Mlist[i])
    plot(day, pReal, pch = 4, cex = 0.5, ylab = expression(pi * xl), yaxt = yaxt,
        xaxt = xaxt)
    lines(day, pMeanBlockMat[i, ], col = c[i])
    legend(x = "topright", legend = paste0("M=", Mlist[i]))
    diffTune = c(diffTune, sum((pMeanBlockMat[i, ] - pReal)^2))
}
```




```
# legend(x='topright', legend = Mlist, col = c, lty = rep(1,7))
diffTune
```

```
## [1] 1.587711 3.421891 1.972059 7.859263 4.566801 1.632534 1.649347 2.183172
```

```
k = seq(3,length(resultList), by=3)
accProbs = c()
for (i in k){
```

```
  accProbs = c(accProbs, resultList[[i]])
}

par(mfrow=c(1,1))
plot(Mlist, time, lty=2, ylim=c(0, max(time)))
lines(Mlist, accProbs*100)
```