TMA4300 Computer Intensive Statistical Methods

Exercise 2, Spring 2022

Martin Tufte, Jim Totland

Problem 1

In this exercise, we take a look at a portion of the Tokyo rainfall data set, a famous data set with daily rainfall data from 1951–1989. The response y_t is the number of days the amount of rainfall exceeded 1mm over the given time period, given by

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

where n_t is the number of years including day $t \in \{1, \cdots, 366\}$. Note that $n_t = 39$ for all t except for $t = 60$, where $n_t = 10$, accounting for February 29th in the leap years. The probability of rain exceeding 1mm on day t is given by $\pi_t(\tau_t)$, which is represented by a latent variable τ_t operating on the logit scale. We assume conditional independence among $y_t|\tau_t$ for all $t \in \{1, \cdots, 366\}$ and let $T = 366$.

a)

We start with exploring the data set by plotting the response as a function of the day t. To get a feel of the trends of the response variable, a monthly mean filter is applied and included. The code for producing this plot is given below.

```
load("rain.rda")
# Applying a monthly mean filter
rain$smooth.rain <- raster::movingFun(rain$n.rain, n=31, fun=mean, circular=TRUE)

ggplot(rain, aes(x = day)) + geom_point(aes(y = n.rain)) +
  geom_line(aes(y=smooth.rain), colour="cadetblue", size=1) +
  xlab("Day in year") + ylab("Number of days rain exceeds 1mm") + theme_minimal()
```

The plot is displayed in Figure 1. From the figure, we observe that there is a pattern, where the rainfall increases towards the middle of the year, where it peaks, and then decreases towards the end of the year. There also seems to be a significant dip for $200 < t < 250$. The mean filter clearly illustrates these patterns. Note that since $n_t = 10$ for $t = 60$, (and not 39 as for all other observations) the apparent anomaly at $t = 60$ may be due to a smaller number of observations. To make the observations more comparable, one could instead have plotted the estimated probabilities $\tilde{\pi}_t = \frac{y_t}{n_t}$, which would correct for the number of years included in each observation.

b)

The likelihood of y_t given τ_t is the probability mass function of a binomial distribution:

$$p(y_t \mid \tau_t) = \binom{n_t}{y_t} \pi(\tau_t)^{y_t} (1-\pi(\tau_t))^{n_t - y_t}, \quad t = 1, \ldots, T. \tag{2}$$

From the conditional independence among the $y_t \mid \tau_t$, it follows that the total likelihood of $\boldsymbol{y}$ given $\boldsymbol{\tau}$ is

$$p(\boldsymbol{y} \mid \boldsymbol{\tau}) = \prod_{t=1}^{T} p(y_t \mid \tau_t) = \prod_{t=1}^{T} \binom{n_t}{y_t} \pi(\tau_t)^{y_t} (1-\pi(\tau_t))^{n_t - y_t}, \tag{3}$$

where $\boldsymbol{y} = \begin{pmatrix} y_1 & \cdots & y_T \end{pmatrix}^\top$ and $\boldsymbol{\tau} = \begin{pmatrix} \tau_1 & \cdots & \tau_T \end{pmatrix}^\top$. We also let $\boldsymbol{\pi} = \begin{pmatrix} \pi(\tau_1) & \cdots & \pi(\tau_T) \end{pmatrix}^\top$.

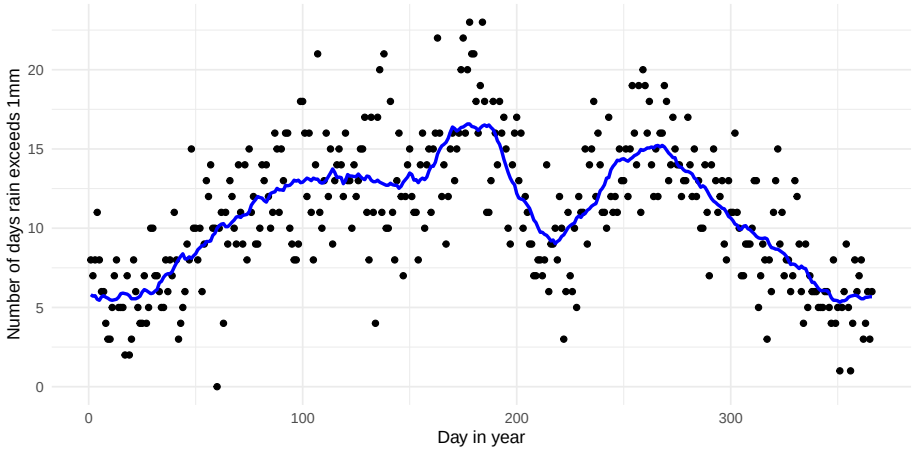


Figure 1: The Tokyo rainfall plotted for each day jointly with a monthly mean filter in blue.

c)

We apply a Bayesian hierarchical model, using a random walk of order 1 to model the trend on a logit scale,

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

where $u_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_u^2)$. We note that $\{\tau_t\}_{t=1}^T$ is a mean-invariant time series. Since $\tau_t - \tau_{t-1} \sim \mathcal{N}(0, \sigma_u^2)$, it follows that $\boldsymbol{u} \mid \sigma_u^2 \sim \mathcal{N}_{T-1}(\mathbf{0}, \sigma_u^2 \mathbf{I})$, where $\boldsymbol{u}$ consists of the $T-1$ consecutive differences in $\boldsymbol{\tau}$. The conditional probability can then be expressed as

$$\begin{aligned} p(\boldsymbol{u} \mid \sigma_u^2) &= \prod_{t=2}^{T} \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left(-\frac{1}{2\sigma_u^2}(u_t)^2\right) \\ &= \frac{1}{\sqrt{(2\pi\sigma_u^2)^{T-1}}} \prod_{t=2}^{T} \exp\left(-\frac{1}{2\sigma_u^2}(\tau_t - \tau_{t-1})^2\right) \\ &= \frac{1}{\sqrt{(2\pi\sigma_u^2)^{T-1}}} \exp\left(-\frac{1}{2\sigma_u^2}\boldsymbol{\tau}^\top \mathbf{Q} \boldsymbol{\tau}\right), \end{aligned}$$

where $\mathbf{Q} \in \mathbb{R}^{T \times T}$ is given by

$$\mathbf{Q} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}.$$

The matrix $\frac{1}{\sigma_u^2}\mathbf{Q}$ then represents a singular precision matrix for the distribution of $\boldsymbol{\tau}$. We will use the notation $p(\boldsymbol{\tau} \mid \sigma_u^2) = p(\boldsymbol{u} \mid \sigma_u^2)$ to represent the density of $\boldsymbol{\tau}$ as a $T-1$ multivariate normal distribution.

In addition, we place an inverse gamma prior on the variance term σ_u^2,

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

for shape α and scale β. The conditional density $p(\sigma_u^2 | \boldsymbol{y}, \boldsymbol{\tau})$ can then be found from the procedure given below.

$$p(\sigma_u^2 \mid \boldsymbol{y}, \boldsymbol{\tau}) = \frac{p(\sigma_u^2, \boldsymbol{y}, \boldsymbol{\tau})}{p(\boldsymbol{y}, \boldsymbol{\tau})} = \frac{p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2) p(\boldsymbol{\tau} \mid \sigma_u^2) p(\sigma_u^2)}{\underbrace{\int p(\boldsymbol{y}, \boldsymbol{\tau} \mid \sigma_u^2) p(\sigma_u^2) \mathrm{d}\sigma_u^2}_{\text{constant w.r.t. } \sigma_u^2}} \propto p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2)\ p(\boldsymbol{\tau} \mid \sigma_u^2)\ p(\sigma_u^2)$$

$$\begin{aligned}
&= \prod_{t=1}^{T} \binom{n_t}{y_t} \pi(\tau_t)^{y_t} (1 - \pi(\tau_t))^{n_t - y_t} \ \times \ \frac{1}{\sqrt{(2\pi\sigma_u^2)^{T-1}}} \exp\left(-\frac{1}{2\sigma_u^2} \boldsymbol{\tau}^\top \mathbf{Q} \boldsymbol{\tau}\right) \\
&\quad \times \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} \exp\left(-\frac{\beta}{\sigma_u^2}\right) \\
&\propto \exp\left(-\frac{1}{\sigma_u^2}\left[\frac{1}{2}\boldsymbol{\tau}^\top \mathbf{Q} \boldsymbol{\tau} + \beta\right]\right) \left(\frac{1}{\sigma_u^2}\right)^{[\alpha + (T-1)/2] + 1}.
\end{aligned}$$

We recognize this as the core of an inverse gamma distribution with shape parameter $\alpha^* = \alpha + (T-1)/2$ and scale parameter $\beta^* = \frac{1}{2}\boldsymbol{\tau}^\top \mathbf{Q} \boldsymbol{\tau} + \beta$. We conclude that

$$\sigma_u^2 \mid \boldsymbol{y}, \boldsymbol{\tau} \sim \text{Inv-Gamma}(\alpha^*, \beta^*).$$

d)

In order to sample from the posterior distribution $p(\boldsymbol{\pi}, \sigma_u^2 | \boldsymbol{y})$ using a Markov Chain Monte Carlo (MCMC) method, we need a proposal distribution. We consider the conditional prior proposal distribution $Q(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)$, where $\mathcal{I} \subseteq \{1, \ldots, T\}$ is a set of time indices and $\boldsymbol{\tau}_{-\mathcal{I}} = \boldsymbol{\tau}_{\{1,\ldots,T\} \setminus \mathcal{I}}$. We want to find an expression for the acceptance probability, $\alpha(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})$, and note in that regard that the current distribution can be written as

$$\begin{aligned}
p(\boldsymbol{\tau} \mid \boldsymbol{y}, \sigma_u^2) &\propto p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2)\ p(\boldsymbol{\tau} \mid \sigma_u^2) \\
&= p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2)\ p(\boldsymbol{\tau}_{\mathcal{I}}, \boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2) \\
&= p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2)\ p(\boldsymbol{\tau}_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)\ p(\boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2).
\end{aligned} \tag{4}$$

Let $\boldsymbol{\tau'}$ be the proposal vector of $\boldsymbol{\tau}$ composed of the proposed values $\boldsymbol{\tau}'_{\mathcal{I}}$ at the indecies $\mathcal{I}$ and the current values $\boldsymbol{\tau}_{-\mathcal{I}}$ at the indecies $-\mathcal{I}$. Then

$$\begin{aligned}
p(\boldsymbol{\tau}' \mid \boldsymbol{y}, \sigma_u^2) &\propto p(\boldsymbol{y} \mid \boldsymbol{\tau}', \sigma_u^2)\ p(\boldsymbol{\tau}' \mid \sigma_u^2) \\
&= p(\boldsymbol{y} \mid \boldsymbol{\tau}', \sigma_u^2)\ p(\boldsymbol{\tau}'_{\mathcal{I}}, \boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2) \\
&= p(\boldsymbol{y} \mid \boldsymbol{\tau}', \sigma_u^2)\ p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)\ p(\boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2).
\end{aligned} \tag{5}$$

In order to satisfy the detailed balance condition, the acceptance probability can be chosen as

$$\alpha(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \min\left\{1,\ \frac{p(\boldsymbol{\tau}' \mid \sigma_u^2, \boldsymbol{y}) \cdot p(\boldsymbol{\tau}_{\mathcal{I}} \mid \boldsymbol{\tau}'_{-\mathcal{I}}, \sigma_u^2)}{p(\boldsymbol{\tau} \mid \sigma_u^2, \boldsymbol{y}) \cdot p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}\right\}. \tag{6}$$

The right term in the minimization in (6) can be simplified using what we found in Equation (4) and (5):

$$\begin{aligned}
\frac{p(\boldsymbol{\tau}' \mid \sigma_u^2, \boldsymbol{y}) \cdot p(\boldsymbol{\tau}_{\mathcal{I}} \mid \boldsymbol{\tau}'_{-\mathcal{I}}, \sigma_u^2)}{p(\boldsymbol{\tau} \mid \sigma_u^2, \boldsymbol{y}) \cdot p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)} &= \frac{p(\boldsymbol{y} \mid \boldsymbol{\tau}', \sigma_u^2)\ p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)\ p(\boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2) \cdot p(\boldsymbol{\tau}_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)}{p(\boldsymbol{y} \mid \boldsymbol{\tau}, \sigma_u^2)\ p(\boldsymbol{\tau}_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)\ p(\boldsymbol{\tau}_{-\mathcal{I}} \mid \sigma_u^2) \cdot p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)} \\
&= \frac{p(\boldsymbol{y} \mid \boldsymbol{\tau}')}{p(\boldsymbol{y} \mid \boldsymbol{\tau})} = \frac{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}'_{\mathcal{I}})\ p(\boldsymbol{y}_{-\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}})}{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}_{\mathcal{I}})\ p(\boldsymbol{y}_{-\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}})} = \frac{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}'_{\mathcal{I}})}{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}_{\mathcal{I}})}.
\end{aligned}$$

Hence, we can write the acceptance probability as

$$\alpha(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \min\left\{1,\ \frac{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}'_{\mathcal{I}})}{p(\boldsymbol{y}_{\mathcal{I}} \mid \boldsymbol{\tau}_{\mathcal{I}})}\right\}. \tag{7}$$

e)

Next, we want to implement an MCMC sampler for the posterior $p(\boldsymbol{\pi}, \sigma_u^2|\boldsymbol{y})$. To achieve this, we use Metropolis-Hastings steps for individual τ_t parameters using the conditional prior $p(\tau_t|\boldsymbol{\tau}_{-t}, \sigma_u^2)$ and Gibbs steps for σ_u^2. It is assumed that $\alpha = 2$ and $\beta = 0.05$, which makes for an informative prior for σ_u^2 placing approximately 95% of the prior mass between 0.01 and 0.25.

We refer to the problem set for an overview of the notation and theory used in this paragraph. In summary, for a multivariate Gaussian variable

$$\boldsymbol{\tau} = \begin{pmatrix} \boldsymbol{\tau}_A \\ \boldsymbol{\tau}_B \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{pmatrix}, \begin{pmatrix} \mathbf{Q}_{AA} & \mathbf{Q}_{AB} \\ \mathbf{Q}_{BA} & \mathbf{Q}_{BB} \end{pmatrix}^{-1} \right),$$

partitioned into two blocks A and B, the conditional mean and conditional precision matrix can be expressed as

$$\begin{aligned} \boldsymbol{\mu}_{A|B} &= -\mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}\boldsymbol{\tau}_B \\ \mathbf{Q}_{A|B} &= \mathbf{Q}_{AA}, \end{aligned}$$

where it is used that the unit vector is a right singular vector of $\mathbf{Q}_{AA}^{-1}\mathbf{Q}_{AB}$ with singular value -1. We note that for $A = t$ and $B = -t$, $\mathbf{Q}_{AA} = \frac{1}{\sigma_u^2}\mathbf{Q}_{[t,t]}$ (where the subscript $[\cdot,\cdot]$ denotes the indices), and $\mathbf{Q}_{AB} = \frac{1}{\sigma_u^2}\mathbf{Q}_{[t,-t]}$. Then it follows that

$$\boldsymbol{\mu}_{t|-t} = \mathbf{Q}_{[t,t]}^{-1}\mathbf{Q}_{[t,-t]}\boldsymbol{\tau}_{-t} = \begin{cases} \tau_2 & , \ t = 1 \\ \frac{1}{2}(\tau_{t-1} + \tau_{t+1}) & , \ 2 \leq t \leq T-1 \\ \tau_{T-1} & , \ t = T \end{cases} \tag{8}$$

$$\mathbf{Q}_{t|-t} = \frac{1}{\sigma_u^2}\mathbf{Q}_{[t,t]} = \begin{cases} \frac{1}{\sigma_u^2} & , \ t = 1 \\ \frac{2}{\sigma_u^2} & , \ 2 \leq t \leq T-1 \\ \frac{1}{\sigma_u^2} & , \ t = T \end{cases}. \tag{9}$$

The acceptance probability is given by Equation (7), and simply amounts to

$$\alpha(\tau_t'|\boldsymbol{\tau}_{-t}, \sigma_u^2, \boldsymbol{y}) = \min\left\{1, \frac{p(y_t|\tau_t')}{p(y_t|\tau_t)}\right\} = \min\left\{1, \left[\frac{\pi(\tau_t')}{\pi(\tau_t)}\right]^{y_t} \cdot \left[\frac{1-\pi(\tau_t')}{1-\pi(\tau_t)}\right]^{n_t - y_t}\right\}, \tag{10}$$

where we used the distribution of $p(y_t \mid \tau_t)$ given in Equation (2). To increase numerical stability, the calculations of the right term in the minimization in Equation (10) is done on log-scale, i.e. by calculating

$$\begin{aligned} &\log\left(\left[\frac{\pi(\tau_t')}{\pi(\tau_t)}\right]^{y_t} \cdot \left[\frac{1-\pi(\tau_t')}{1-\pi(\tau_t)}\right]^{n_t - y_t}\right) \\ &= y_t \cdot \Big[\log(e^{-\tau_t} + 1) - \log(e^{-\tau_t'} + 1)\Big] + (n_t - y_t) \cdot \Big[\log(e^{\tau_t} + 1) - \log(e^{\tau_t'} + 1)\Big] \\ &= y_t(\tau_t' - \tau_t) + n_t \log\left(\frac{1 + e^{\tau_t}}{1 + e^{\tau_t'}}\right). \end{aligned}$$

To increase computational efficiency, a couple of optimizations are done. The first is concerning the Metropolis-Hastings steps. For each iteration, we precompute the probabilities used for each $\tau_1, \ldots, \tau_T$. In addition, to avoid unnecessary matrix multiplications, the reduced expressions in Equation (8) and (9) are used. The second is concerning the Gibbs step, where we used that $\boldsymbol{\tau}^\top \mathbf{Q} \boldsymbol{\tau} = \sum_{i=2}^{T}(\tau_i - \tau_{i-1})^2$ when computing the parameter β^* for the conditional distribution of σ_u^2.

Our MCMC sampler is given in the first code block below. The second block of code runs the MCMC algorithm with single site updates and conducts all the necessary computations for this problem. This includes the calculations of computation times, acceptance rates and estimated autocorrelation functions.

```r
mcmc.single <- function(num.iterations, initial.tau, initial.sigma2){
  ### Initialization
  T            <- length(initial.tau)
  tau          <- matrix(NA, nrow = (num.iterations+1), ncol = T)
  tau[1, ]     <- initial.tau
  sigma2       <- c(initial.sigma2, rep(NA, num.iterations))
  num.accepted <- 0
  alpha.star   <- 2.00 + (T-1)/2
  beta         <- 0.05
  root2        <- sqrt(2)

  ### Iterations
  for(i in 2:(num.iterations+1)){

    ### Metropolis-Hastings steps for tau
    tau[i, ] <- tau[i-1, ]
    # Precompute probabilities
    U <- runif(T)
    Z.sigma <- sqrt(sigma2[i-1]) * rnorm(T)
    # t = 1
    tau.proposal <- tau[i,2] + Z.sigma[1]
    tau.current  <- tau[i,1]
    log.acceptance <- y[1]*(tau.proposal - tau.current) +
      n[1]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
    if( U[1] < exp(log.acceptance) ){
      tau[i,1] = tau.proposal
      num.accepted <- num.accepted+1
    }
    # t = 2:T-1
    for(t in 2:(T-1)){
      tau.proposal <- (tau[i,t-1] + tau[i,t+1])/2 + Z.sigma[t]/root2
      tau.current  <- tau[i,t]
      log.acceptance <- y[t]*(tau.proposal - tau.current) +
        n[t]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
      if( U[t] < exp(log.acceptance) ){
        tau[i, t] = tau.proposal
        num.accepted <- num.accepted+1
      }
    }
    # t = T
    tau.proposal <- tau[i,T-1] + Z.sigma[T]
    tau.current  <- tau[i,T]
    log.acceptance <- y[T]*(tau.proposal - tau.current) +
      n[T]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
    if(U[T] < exp(log.acceptance) ){
      tau[i, T] = tau.proposal
      num.accepted <- num.accepted+1
    }

    ### Gibbs step for sigma^2
    sigma2[i] <- 1/rgamma(1, shape = alpha.star, rate = 0.5*sum(diff(tau[i,])^2) + beta)
  }

  ### Return tau, sigma^2 and acceptance rates for tau
  list(tau = tau, sigma2 = sigma2, acceptance.rate = num.accepted/(num.iterations*T))
}
```

```r
# Script for problem 1e)
y <- rain$n.rain  # response
n <- rain$n.years # number of years
T <- length(y)    # days in a year (366)

# Parameters for the prior of sigma^2
alpha <- 2.00
beta <- 0.05


# Run the MCMC for 50000 iterations
set.seed(4300)
num.iter    <- 50000
init.tau    <- runif(T,-3,0)
init.sigma2 <- 0.01
ptm <- proc.time()
mcmc <- mcmc.single(num.iter, init.tau, init.sigma2, y, n, alpha, beta)
elapsed.time <- (proc.time() - ptm)[3]


# Elapsed time
print(paste("Time elapsed for", num.iter, "iterations is", round(elapsed.time,8), "seconds."))
# Acceptance rate
print(paste("The acceptance rate is", round(mcmc$acceptance.rate,8) ))

## Traceplots, histograms, and sample autocorrelation functions
mcmc.data.all <- data.frame("x"         = 1:(num.iter+1),
                            "sigma.vec" = mcmc$sigma2,
                            "pi_1"      = sigm(mcmc$tau[,1]),
                            "pi_201"    = sigm(mcmc$tau[,201]),
                            "pi_366"    = sigm(mcmc$tau[,366])
                            )

burn.in = 500
if(burn.in){
  mcmc.data <- mcmc.data.all[-(1:burn.in),]
} else{
  mcmc.data <- mcmc.data.all
}

max.lag <- 50
mcmc.corr <- data.frame("lag"       = 0:max.lag,
                        "sigma.vec" = sacf(mcmc$sigma2, max.lag)$rho.hat,
                        "pi_1"      = sacf(sigm(mcmc$tau[,1]), max.lag)$rho.hat,
                        "pi_201"    = sacf(sigm(mcmc$tau[,201]), max.lag)$rho.hat,
                        "pi_366"    = sacf(sigm(mcmc$tau[,366]), max.lag)$rho.hat)


# Traceplots, histograms, and sample autocorrelation for sigma^2
traceplot.sigma <- ggplot(mcmc.data.all, aes(x=x,y=sigma.vec)) +
  geom_line() + xlab("Iterations") + ylab(expression(sigma[u]^2)) +
  ylim(0,0.05) + theme_minimal()
traceplot.sigma
ggsave("./figures/traceplot_sigma2.pdf", plot = traceplot.sigma, height = 4.0, width = 8.0)

histogram.sigma <- ggplot(mcmc.data, aes(x=sigma.vec)) +
```

```
  geom_histogram(binwidth=0.0005) + xlab(expression(sigma[u]^2)) + ylab("Count") +
  theme_minimal()
histogram.sigma
ggsave("./figures/histogram_sigma2.pdf", plot = histogram.sigma, height = 4.0, width = 8.0)

correlation.sigma <- ggplot(mcmc.corr, aes(x=lag, y=sigma.vec)) +
  geom_hline(aes(yintercept = 0)) +
  geom_segment(mapping = aes(xend = lag, yend = 0)) + geom_point(size=0.3) +
  xlab("Lag") + ylab("Correlation") + theme_minimal()
correlation.sigma
ggsave("./figures/correlation_sigma2.pdf", plot = correlation.sigma, height = 4.0, width = 8.0)


# pi_1, pi_201, pi_366
traceplot.tau <- ggplot(mcmc.data.all, aes(x=x)) +
  geom_line(aes(y=pi_1, colour="tau_1"),size=0.25, alpha=0.4) +
  geom_line(aes(y=pi_201, colour="tau_201"),size=0.25, alpha=0.4) +
  geom_line(aes(y=pi_366, colour="tau_366"),size=0.25, alpha=0.4) +
  scale_color_manual(name="", values=c("tau_1"="red", "tau_201"="blue", "tau_366"="green"),
                     l= expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab("Iterations") + ylab(" ") + theme_minimal()
traceplot.tau
ggsave("./figures/traceplot_tau.pdf", plot = traceplot.tau, height = 4.0, width = 8.0)

histogram.tau <- ggplot(mcmc.data) +
  geom_histogram(aes(x=pi_1, fill="tau_1"), binwidth=0.005, alpha=0.6) +
  geom_histogram(aes(x=pi_201, fill="tau_201"), binwidth=0.005, alpha=0.6) +
  geom_histogram(aes(x=pi_366, fill="tau_366"), binwidth=0.005, alpha=0.6) +
  scale_fill_manual(name = " ", values = c("tau_1" = "red", "tau_201" = "blue", "tau_366" = "green"),
                    label = expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab(" ") + ylab("Count") + theme_minimal()
histogram.tau
ggsave("./figures/histogram_tau.pdf", plot = histogram.tau, height = 4.0, width = 8.0)

correlation.tau <- ggplot(mcmc.corr, aes(x=lag)) +
  geom_hline(aes(yintercept = 0)) +
  geom_line(aes(y=pi_1, colour="tau_1"),size=0.25) +
  geom_point(aes(y=pi_1, colour="tau_1"), size =0.3) +
  geom_line(aes(y=pi_201, colour="tau_201"),size=0.25) +
  geom_point(aes(y=pi_201, colour="tau_201"), size =0.3) +
  geom_line(aes(y=pi_366, colour="tau_366"),size=0.25) +
  geom_point(aes(y=pi_366, colour="tau_366"), size =0.3) +
  xlab("Lag") + ylab("Correlation") +
  scale_color_manual(name = "", values = c("tau_1" = "red", "tau_201" = "blue", "tau_366" = "green"),
                     label = expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab("Lag") + ylab("Autcorrelation") + theme_minimal()
correlation.tau
ggsave("./figures/correlation_tau.pdf", plot = correlation.tau, height = 4.0, width = 8.0)


# Plot predictions of pi
pi.preds <- plot.preds(mcmc$tau)
pi.preds
ggsave("./figures/pi_preds.pdf", plot = pi.preds, height = 4.0, width = 8.0)
```

```r
### Calculate statistics for tau_1, tau_201, tau_366 and sigma
tau.idx <- c(1, 201, 366)
pi.df <- plot.preds(mcmc$tau, plot = FALSE)


tau.table <- data.frame(idx = tau.idx,
                        pi = pi.df$pi[tau.idx],
                        lower = pi.df$lower[tau.idx],
                        upper = pi.df$upper[tau.idx])
print(tau.table)


sigma.q <- quantile(mcmc$sigma2, probs = c(0.025, 0.975))
sigma.table = c(pred = mean(mcmc$sigma2), lower = sigma.q[1], upper = sigma.q[2])
print(sigma.table)

# Estimated ESS
effectiveSize(as.mcmc(mcmc.data))
```

By use of `proc.time()`, we calculated the computation time to be 35.8 seconds. The acceptance for individual τ_t was, on average, approximately 0.9174. Central estimates of $\pi(\tau_1), \pi(\tau_{201}), \pi(\tau_{366})$, and σ_u^2 together with 95 % credible intervals and estimated effective sample sizes (ESS) are given in Table 1. The estimates are calculated with burn-in period of 500 (i.e. we removed the first 500 samples). The burn-in period is based on the traceplots in Figures 3a and 4a, which seem to illustrate a rather quick convergence.

Table 1: Prediction and credible intervals and estimated ESS for some selected entries of $\boldsymbol{\pi}$ and σ_u^2, generated by our MCMC-algorithm.

Variable	Prediction	2.5% quantile	97.25% quantile	Estiamted ESS
$\pi(\tau_1)$	0.1834	0.1330	0.2463	2596
$\pi(\tau_{201})$	0.3329	0.2788	0.3912	3129
$\pi(\tau_{366})$	0.1315	0.09192	0.1843	2165
σ_u^2	0.01015	0.00594	0.01623	1208

The predictions of $\boldsymbol{\pi}$ generated by the MCMC-algorithm are displayed in Figure 2. That is, the central estimates and equi-tailed 95% credible intervals, along with y_t/n_t from the data set. The traceplots, histograms and sample autocorrelations for σ_u^2 are displayed in Figure 3 and for the values of τ_t in Figure 4. The traceplots indicates that the Markov chain has converged, as the trace seems to fluctuate around a mean value and the fluctuation seems to be random.

From the histograms in Figures 4b and 3b, we see that the sampler of σ_u^2 appears to have converged to an inverse gamma distribution, and the τ_t's also seem to have converged to a distribution.

The samples are not uncorrelated, as indicated by the estimated sample autocorrelations in Figures 3c and 4c. For both the samples of $\pi(\tau_t)$ and σ_u^2, the autocorrelation decays and seems to approach zero, though there is still some positive autocorrelation for lags larger than 50 (maybe excluding $\pi(\tau_{201})$). This can be explained by the fact that the τ_t's are updated individually conditional on the previous sample of the parameters. Thus, we may only explore a limited neighborhood in the parameter space, making consecutive samples highly correlated.

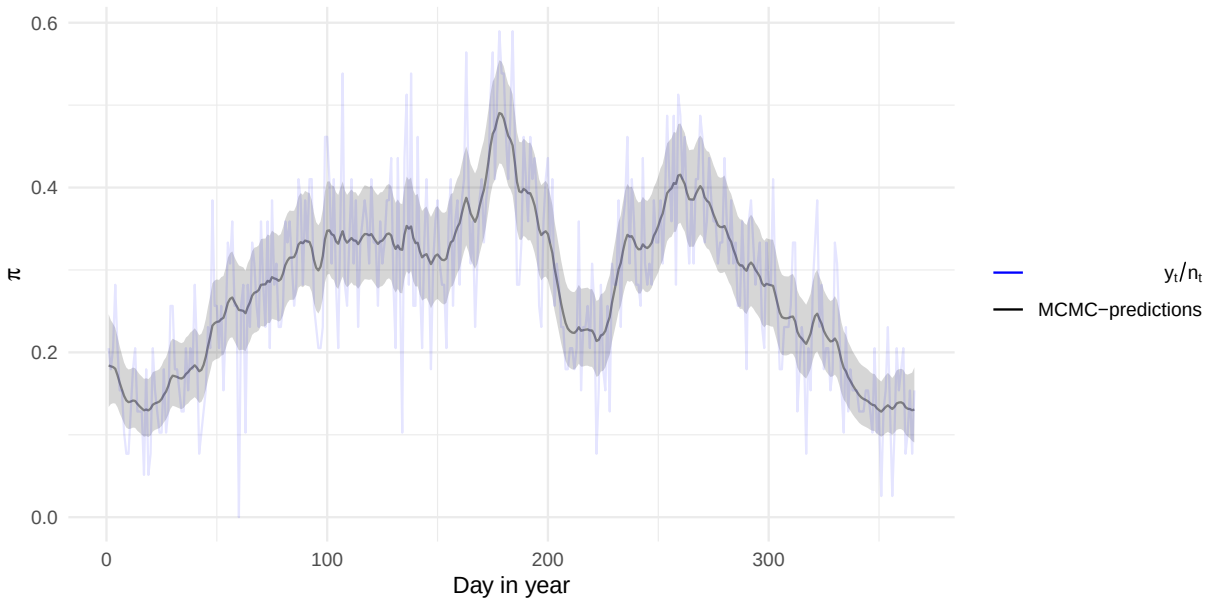


Figure 2: The MCMC-predictions of $\boldsymbol{\pi}$ (black) with associated 95% credible intervals (gray area). The fractions y_t/n_t are displayed in blue.

f)

In problem **e)** we used single site updating of individual τ_t. Now, we instead use a conditional prior proposal involving $p(\boldsymbol{\tau}_{(a,b)} \mid \boldsymbol{\tau}_{-(a,b)}, \sigma_u^2)$, where $\boldsymbol{\tau}_{(a,b)} = \begin{pmatrix} \tau_a & \cdots & \tau_b \end{pmatrix}^\top$ choosing intervals of length M, assuming $1 < M < T$. Generalizing Equation 10, which was only valid for a single index t, let $\mathcal{I} \subseteq \{1, \cdots, T\}$ be a general index set. The acceptance probability can then be expressed as

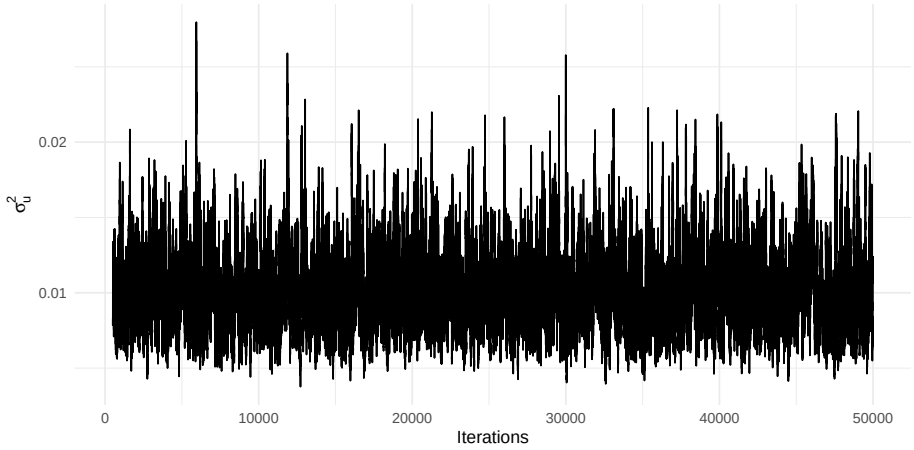
$$\begin{aligned}
\alpha(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) &= \min\left\{1, \prod_{t\in\mathcal{I}} \left[\frac{\pi(\tau'_t)}{\pi(\tau_t)}\right]^{y_t} \cdot \left[\frac{1-\pi(\tau'_t)}{1-\pi(\tau_t)}\right]^{n_t - y_t}\right\} \\
&= \exp\min\left\{0, \sum_{t\in\mathcal{I}} y_t \cdot \Big[\log(e^{-\tau_t}+1) - \log(e^{-\tau'_t}+1)\Big] + (n_t - y_t) \cdot \Big[\log(e^{\tau_t}+1) - \log(e^{\tau'_t}+1)\Big]\right\} \\
&= \exp\min\left\{0, \sum_{t\in\mathcal{I}} \left[y_t(\tau'_t - \tau_t) + n_t \log\left(\frac{1+e^{\tau_t}}{1+e^{\tau'_t}}\right)\right]\right\},
\end{aligned}$$

and the proposal density $Q(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = p(\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)$ has distribution given by

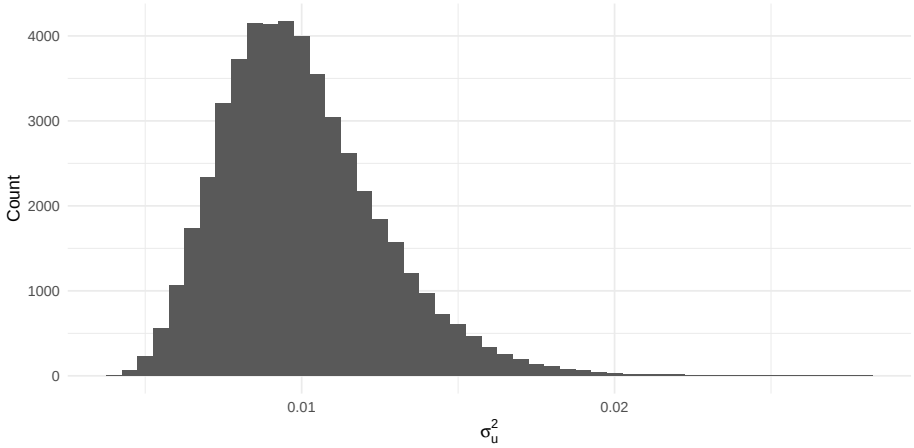
$$\boldsymbol{\tau}'_{\mathcal{I}} \mid \boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2 \sim \mathcal{N}_{|\mathcal{I}|}\left(-\mathbf{Q}^{-1}_{[\mathcal{I},\mathcal{I}]}\mathbf{Q}_{[\mathcal{I},-\mathcal{I}]}\boldsymbol{\tau}_{-\mathcal{I}},\ \sigma_u^2\mathbf{Q}^{-1}_{[\mathcal{I},\mathcal{I}]}\right).$$

When using intervals of length M, there are potentially three cases for $\mathbf{Q}^{-1}_{[\mathcal{I},\mathcal{I}]}$. These will not change during the MCMC iterations, so it is beneficial to precompute these quantities beforehand. The first case is when $a = 1$, then the index set is given as $\mathcal{I} = (1, \cdots, M)$. One can show that

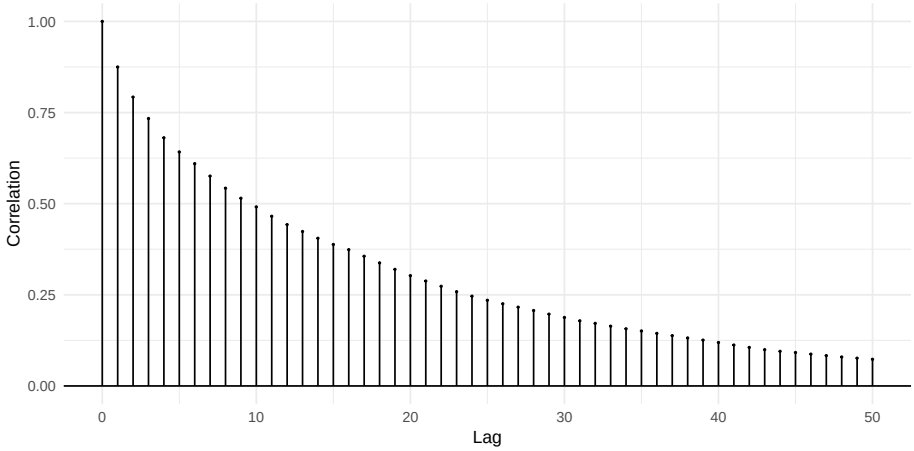
$$\mathbf{Q}^{-1}_{[(1,M),(1,M)]} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}^{-1} = \begin{pmatrix} M & M-1 & M-2 & \cdots & 1 \\ M-1 & M-1 & M-2 & \cdots & 1 \\ M-2 & M-2 & M-2 & & 1 \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & 1 & 1 & & 1 \end{pmatrix}.$$



(a) Traceplot.



(b) Histogram.

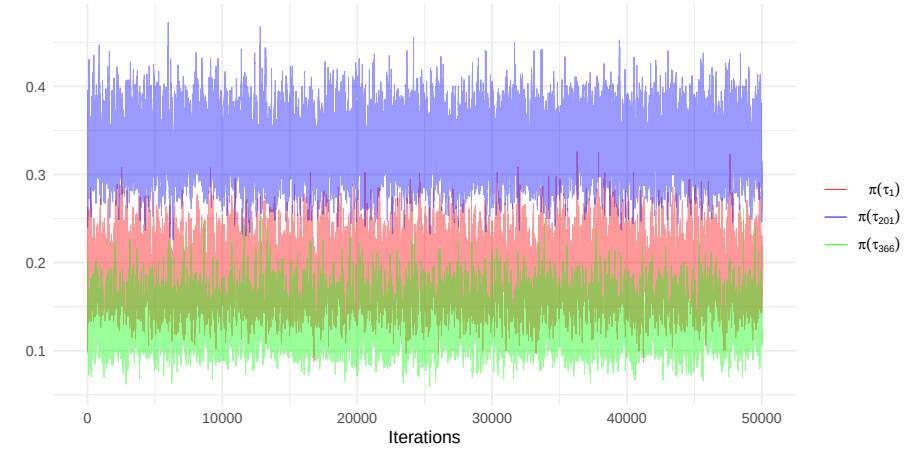


(c) Autocorrelation.

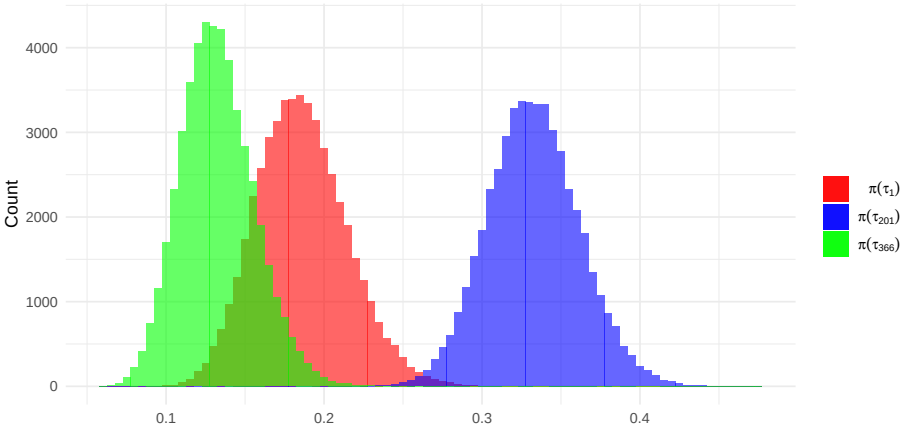
Figure 3: Traceplot (a), histogram (b) and autocorrelation (c) for σ_u^2 using the MCMC algorithm with single site updates of $\boldsymbol{\tau}$. The histogram and autocorrelation plots are computed with a burn-in period of 500 samples.

Next, it follows that the conditional mean $\boldsymbol{\mu}_{(1,M)} = -\mathbf{Q}_{[(1,M),(1,M)]}^{-1}\mathbf{Q}_{[(1,M),-(1,M)]}\boldsymbol{\tau}_{-\mathcal{I}}$ for this case is

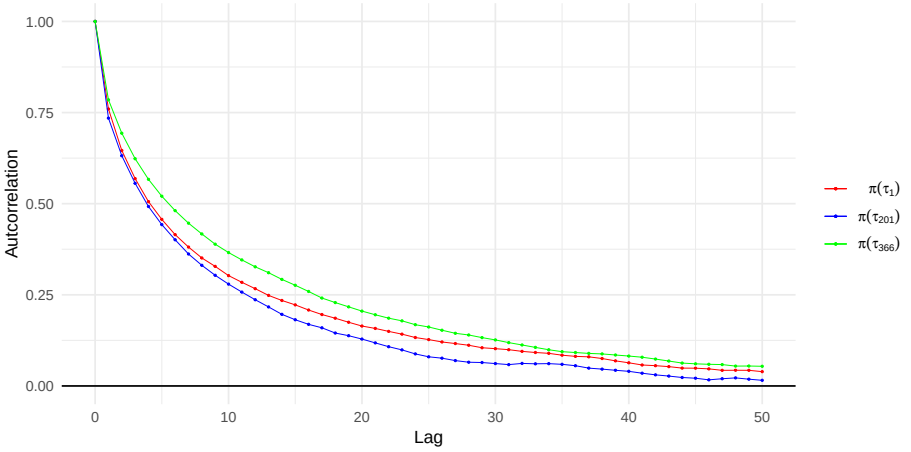
$$\boldsymbol{\mu}_{(1,M)} = -\begin{pmatrix} M & M-1 & M-2 & \cdots & 1 \\ M-1 & M-1 & M-2 & \cdots & 1 \\ M-2 & M-2 & M-2 & & 1 \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & 1 & 1 & & 1 \end{pmatrix} \underbrace{\begin{pmatrix} & & & \\ & & & \\ & & & \\ -1 & & & \end{pmatrix}}_{M\times(T-M)} \begin{pmatrix} \tau_{M+1} \\ \tau_{M+2} \\ \vdots \\ \tau_{T-1} \\ \tau_T \end{pmatrix} = \begin{pmatrix} \tau_{M+1} \\ \tau_{M+1} \\ \vdots \\ \tau_{M+1} \\ \tau_{M+1} \end{pmatrix}.$$



(a) Traceplot.



(b) Histogram.



(c) Autocorrelation.

Figure 4: Traceplot (a), histogram (b) and autocorrelation (c) for $\pi(\tau_1)$ in red, $\pi(\tau_{201})$ in blue and $\pi(\tau_{366})$ in green using the single site MCMC algorithm. The histogram and autocorrelation plots are computed with a burn-in period of 500 samples.

The second case is when $a > 1$ and $b < T$. Then

$$\mathbf{Q}^{-1}_{[(a,b),(a,b)]} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}^{-1} = \frac{1}{M+1} \begin{pmatrix} M & \cdots & 3 & 2 & 1 \\ \vdots & 2(M-1) & \cdots & 2\cdot 2 & 2\cdot 1 \\ 3 & \vdots & 3(M-2) & \cdots & 3\cdot 1 \\ 2 & 2\cdot 2 & \vdots & \ddots & \vdots \\ 1 & 2\cdot 1 & 3\cdot 1 & \cdots & M\cdot 1 \end{pmatrix}.$$

Next, it follows that the conditional mean $\boldsymbol{\mu}_{(a,b)} = -\mathbf{Q}^{-1}_{[(1,M),(1,M)]}\mathbf{Q}_{[(1,M),-(1,M)]}$ for the second case is

$$\boldsymbol{\mu}_{(a,b)} = \frac{-1}{M+1}\begin{pmatrix} M & \cdots & 3 & 2 & 1 \\ \vdots & 2(M-1) & \cdots & 2\cdot 2 & 2\cdot 1 \\ 3 & \vdots & 3(M-2) & \cdots & 3\cdot 1 \\ 2 & 2\cdot 2 & \vdots & \ddots & \vdots \\ 1 & 2\cdot 1 & 3\cdot 1 & \cdots & M\cdot 1 \end{pmatrix} \underbrace{\begin{pmatrix} -1 & & \\ & & \\ & & -1 \end{pmatrix}}_{M\times(T-M)} \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_{a-1} \\ \tau_{b+1} \\ \vdots \\ \tau_T \end{pmatrix} = \frac{1}{M+1}\begin{pmatrix} M\tau_{a-1}+\tau_{b+1} \\ (M-1)\tau_{a-1}+2\tau_{b+1} \\ \vdots \\ 2\tau_{a-1}+(M-1)\tau_{b+1} \\ \tau_{a-1}+M\tau_{b+1} \end{pmatrix}.$$

The third case is when $b = T$. Here we must take into consideration that if M does not divide T, this matrix will be smaller than the other ones. Finally, this gives

$$\mathbf{Q}^{-1}_{[(a,T),(a,T)]} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & & 1 & 1 & 1 \\ \vdots & \ddots & & \vdots & \vdots \\ 1 & & M'-2 & M'-2 & M'-2 \\ 1 & \cdots & M'-2 & M'-1 & M'-1 \\ 1 & \cdots & M'-2 & M'-1 & M' \end{pmatrix},$$

where $1 \leq M' \leq M$ depending on how M divides T. Similarly to the first case, this gives a conditional mean of

$$\boldsymbol{\mu}_{(a,T)} = \begin{pmatrix} \tau_{a-1} \\ \tau_{a-1} \\ \vdots \\ \tau_{a-1} \\ \tau_{a-1} \end{pmatrix}.$$

The following code chunk implements `mcmc.block`, an MCMC sampler with block updates for $\boldsymbol{\tau}$ and Gibbs steps for σ_u^2. The second code block runs the MCMC algorithm with blocking and conducts all the necessary computations for this problem. This includes the calculations to find optimal values of M, computation times, acceptance rates and estimated autocorrelations.

```
mcmc.block <- function(num.iterations, initial.tau, initial.sigma2, y, n, M,
alpha=2.00, beta=0.05){
  if(M==1) return( mcmc.single(num.iterations, initial.tau, initial.sigma2, y, n,
  alpha=2.00, beta=0.05) )

  ### Initialization
  T            <- length(initial.tau)
  tau          <- matrix(NA, nrow = (num.iterations+1), ncol = T)
  tau[1, ]     <- initial.tau
  sigma2       <- c(initial.sigma2, rep(NA, num.iterations))
  num.accepted <- 0
  alpha.star   <- alpha + (T-1)/2
  root2        <- sqrt(2)


  ### Precomputing (assuming M < T)
  indecies <- get.indecies(T,M)
  num.blocks <- dim(indecies)[1]
  M.last <- T - indecies[num.blocks,1]+1

  # Create Q matrix
  Q <- diag(c(1, rep(2,T-2),1))
  Q[abs(row(Q) - col(Q))==1] <- -1
```

```
# Q.AA for the three different blocks
Q.AA.inv1 <- solve( Q[1:M, 1:M] )
Q.AA.inv2 <- solve( Q[2:(M+1), 2:(M+1)] )
Q.AA.inv3 <- solve( Q[indecies[num.blocks,1]:T, indecies[num.blocks,1]:T] )


### Iterations
for(i in 2:(num.iter+1)){

  ### Metropolis-Hastings steps for tau
  tau[i, ] <- tau[i-1, ]
  sigma2[i] <- sigma2[i-1]
  Q.AA.inv1.sigma2 <- sigma2[i]*Q.AA.inv1
  Q.AA.inv2.sigma2 <- sigma2[i]*Q.AA.inv2
  Q.AA.inv3.sigma2 <- sigma2[i]*Q.AA.inv3

  # precompute probabilities
  U <- runif(num.blocks)

  # I = (1:M)
  I = 1:M
  tau.proposal <- mvrnorm(1, mu=rep(tau[i,M+1],M), Sigma=Q.AA.inv1.sigma2)
  tau.current  <- tau[i,I]
  log.acceptance <- y[I]*(tau.proposal - tau.current) +
    n[I]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
  if( U[1] < exp(sum(log.acceptance)) ){
    tau[i,I] = tau.proposal
    num.accepted <- num.accepted+M
  }
  # I = (M+1:2M), ...
  for(j in 2:(num.blocks-1)){

    a <- indecies[j,1]
    b <- indecies[j,2]
    I = a:b
    tau.proposal <- mvrnorm(1, mu=linspace(tau[i,a-1], tau[i,b+1], M+2)[2:(M+1)],
    Sigma=Q.AA.inv2.sigma2)
    tau.current  <- tau[i,I]
    log.acceptance <- y[I]*(tau.proposal - tau.current) +
      n[I]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
    if( U[j] < exp(sum(log.acceptance)) ){
      tau[i,I] = tau.proposal
      num.accepted <- num.accepted+M
    }
  }
  # I = ((T-M.last+1):T)
  I = indecies[num.blocks,1]:T
  tau.proposal <- mvrnorm(1, mu=rep(tau[i,T-M], M.last), Sigma=Q.AA.inv3.sigma2)
  tau.current  <- tau[i,I]
  log.acceptance <- y[I]*(tau.proposal - tau.current) +
    n[I]*log( (1+exp(tau.current))/(1+exp(tau.proposal)) )
  if( U[num.blocks] < exp(sum(log.acceptance)) ){
    tau[i,I] = tau.proposal
    num.accepted <- num.accepted+M.last
  }
```

```
    ### Gibbs step for sigma^2
    sigma2[i] <- 1/rgamma(1, shape = alpha.star, rate = 0.5*sum(diff(tau[i,])^2) + beta)
  }

  ### Return tau, sigma^2 and acceptance rates for tau
  list(tau = tau, sigma2 = sigma2, acceptance.rate = num.accepted/(num.iterations*T))
}
```

```
# Script for problem 1f)
y <- rain$n.rain  # response
n <- rain$n.years # number of years
T <- length(y)    # days in a year (366)

# Parameters for the prior of sigma^2
alpha <- 2.00
beta <- 0.05

# Blocking parameter
M <- 10

# Run the MCMC for 50000 iterations
set.seed(4300)
num.iter    <- 50000
init.tau    <- runif(T,-3,0)
init.sigma2 <- 0.01
ptm <- proc.time()
mcmc <- mcmc.block(num.iter, init.tau, init.sigma2, y, n, M, alpha, beta)
elapsed.time <- (proc.time() - ptm)[3]

# Elapsed time
print(paste("Time elapsed for", num.iter, "iterations is", round(elapsed.time,8), "seconds."))
# Acceptance rate
print(paste("The acceptance rate is", round(mcmc$acceptance.rate,8) ))

## Traceplots, histograms, and sample autocorrelation functions
mcmc.data.all <- data.frame("x"         = 1:(num.iter+1),
                            "sigma.vec" = mcmc$sigma2,
                            "pi_1"      = sigm(mcmc$tau[,1]),
                            "pi_201"    = sigm(mcmc$tau[,201]),
                            "pi_366"    = sigm(mcmc$tau[,366])
)

burn.in = 500
if(burn.in){
  mcmc.data <- mcmc.data.all[-(1:burn.in),]
} else{
  mcmc.data <- mcmc.data.all
}

max.lag <- 50
mcmc.corr <- data.frame("lag"       = 0:max.lag,
                        "sigma.vec" = sacf(mcmc$sigma2, max.lag)$rho.hat,
                        "pi_1"      = sacf(sigm(mcmc$tau[,1]), max.lag)$rho.hat,
                        "pi_201"    = sacf(sigm(mcmc$tau[,201]), max.lag)$rho.hat,
                        "pi_366"    = sacf(sigm(mcmc$tau[,366]), max.lag)$rho.hat
```

```
)

# Traceplots, histograms, and sample autocorrelation for sigma^2
traceplot.sigma <- ggplot(mcmc.data.all, aes(x=x,y=sigma.vec)) +
  geom_line() + xlab("Iterations") + ylab(expression(sigma[u]^2)) +
  ylim(0, 0.05) + theme_minimal()
traceplot.sigma
ggsave("./figures/traceplot_sigma2_block.pdf", plot = traceplot.sigma, height = 4.0, width = 8.0)

histogram.sigma <- ggplot(mcmc.data, aes(x=sigma.vec)) +
  geom_histogram(binwidth=0.0005) + xlab(expression(sigma[u]^2)) + ylab("Count") +
  theme_minimal()
histogram.sigma
ggsave("./figures/histogram_sigma2_block.pdf", plot = histogram.sigma, height = 4.0, width = 8.0)

correlation.sigma <- ggplot(mcmc.corr, aes(x=lag, y=sigma.vec)) +
  geom_hline(aes(yintercept = 0)) +
  geom_segment(mapping = aes(xend = lag, yend = 0)) + geom_point(size=0.3) +
  xlab("Lag") + ylab("Correlation") + theme_minimal()
correlation.sigma
ggsave("./figures/correlation_sigma2_block.pdf", plot = correlation.sigma, height = 4.0, width = 8.0)


# pi_1, pi_201, pi_366
traceplot.tau <- ggplot(mcmc.data, aes(x=x)) +
  geom_line(aes(y=pi_1, colour="tau_1"),size=0.25, alpha=0.4) +
  geom_line(aes(y=pi_201, colour="tau_201"),size=0.25, alpha=0.4) +
  geom_line(aes(y=pi_366, colour="tau_366"),size=0.25, alpha=0.4) +
  scale_color_manual(name="", values=c("tau_1"="red", "tau_201"="blue", "tau_366"="green"),
                     labels = expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab("Iterations") + ylab(" ") + theme_minimal()
traceplot.tau
ggsave("./figures/traceplot_tau_block.pdf", plot = traceplot.tau, height = 4.0, width = 8.0)

histogram.tau <- ggplot(mcmc.data) +
  geom_histogram(aes(x=pi_1, fill="tau_1"), binwidth=0.005, alpha=0.6) +
  geom_histogram(aes(x=pi_201, fill="tau_201"), binwidth=0.005, alpha=0.6) +
  geom_histogram(aes(x=pi_366, fill="tau_366"), binwidth=0.005, alpha=0.6) +
  scale_fill_manual(name = " ", values = c("tau_1" = "red", "tau_201" = "blue", "tau_366" = "green"),
                    labels = expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab(" ") + ylab("Count") + theme_minimal()
histogram.tau
ggsave("./figures/histogram_tau_block.pdf", plot = histogram.tau, height = 4.0, width = 8.0)

correlation.tau <- ggplot(mcmc.corr, aes(x=lag)) +
  geom_hline(aes(yintercept = 0)) +
  geom_line(aes(y=pi_1, colour="tau_1"),size=0.25) +
  geom_point(aes(y=pi_1, colour="tau_1"), size =0.3) +
  geom_line(aes(y=pi_201, colour="tau_201"),size=0.25) +
  geom_point(aes(y=pi_201, colour="tau_201"), size =0.3) +
  geom_line(aes(y=pi_366, colour="tau_366"),size=0.25) +
  geom_point(aes(y=pi_366, colour="tau_366"), size =0.3) +
  xlab("Lag") + ylab("Correlation") +
  scale_color_manual(name = "", values = c("tau_1" = "red", "tau_201" = "blue", "tau_366" = "green"),
                     label = expression(pi(tau[1]),pi(tau[201]),pi(tau[366]))) +
  xlab("Lag") + ylab("Autcorrelation") + theme_minimal()
```

```r
correlation.tau
ggsave("./figures/correlation_tau_block.pdf", plot = correlation.tau, height = 4.0, width = 8.0)


# Plot predictions of pi
pi.preds <- plot.preds(mcmc$tau)
pi.preds
ggsave("./figures/pi_preds_block.pdf", plot = pi.preds, height = 4.0, width = 8.0)


### Calculate statistics for tau_1, tau_201, tau_366 and sigma
tau.idx <- c(1, 201, 366)
pi.df <- plot.preds(mcmc$tau, plot = FALSE)

tau.table <- data.frame(idx = tau.idx,
                        pi = pi.df$pi[tau.idx],
                        lower = pi.df$lower[tau.idx],
                        upper = pi.df$upper[tau.idx])
print(tau.table)


sigma.q <- quantile(mcmc$sigma2, probs = c(0.025, 0.975))
sigma.table = c(pred = mean(mcmc$sigma2), lower = sigma.q[1], upper = sigma.q[2])
print(sigma.table)

# Estimated ESS
effectiveSize(as.mcmc(mcmc.data))

### Estimate computation time/acceptance rate as function of M
Ms               <- 1:100
elapsed.times    <- numeric(100)
acceptance.rates <- numeric(100)

for(M in Ms){
  # Run MCMC
  num.iter <- 1000
  set.seed(4300)
  ptm <- proc.time()
  mcmc <- mcmc.block(num.iter, init.tau, init.sigma2, y, n, M, alpha, beta)
  elapsed.time <- (proc.time() - ptm)[3]
  print(paste("Time elapsed for", num.iter, "iterations is", round(elapsed.time,8), "seconds." ))
  ?runif
  # Calculate acceptance rate
  print(paste("The acceptance rate for M=", M, "is", round(mcmc$acceptance.rate,8) ))

  elapsed.times[M] <- elapsed.time
  acceptance.rates[M] <- mcmc$acceptance.rate
}

data.M <- data.frame("M" = Ms,
                     "elapsed.times" = elapsed.times,
                     "acceptance.rates" = acceptance.rates)

plot(elapsed.times, type='l')
plot(acceptance.rates, type='l')
```

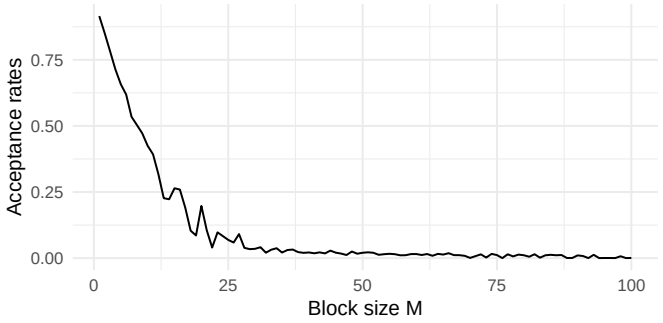
```
elapsed.times.plot <- ggplot(data.M, aes(x=M, y=elapsed.times)) +
  geom_line() + xlab("Block size M") + ylab("Computation time [seconds]") +
  theme_minimal()
elapsed.times.plot
ggsave("./figures/elapsed.times.plot.pdf", plot = elapsed.times.plot, height = 2.5, width = 5.0)

acceptance.rates.plot <- ggplot(data.M, aes(x=M, y=acceptance.rates)) +
  geom_line() + xlab("Block size M") + ylab("Acceptance rates") +
  theme_minimal()
acceptance.rates.plot
ggsave("./figures/acceptance.rates.plot.pdf", plot = acceptance.rates.plot, height = 2.5, width = 5.0)
```

To test whether incorporating a block step over the $\boldsymbol{\tau}_{(a,b)}$ parameters will improve the efficiency of our MCMC sampler, we simulated for 1000 iterations using $M = 1, 2, \cdots, 100$. Figure 5 displays the acceptance rates and the computation times. Note that for $M = 1$, we used the normal MCMC implementation given in Problem 1**e)**. From subplot (a) it is clear that the acceptance rates decrease with increasing values of M. This is expected, as the proposal constitutes an increasing number of new variables when M increases. E.g. for $M = 10$ the new proposal updates 10 variables in $\boldsymbol{\tau}$ at a time. This could lead to the sampler proposing a less probable proposal, and hence the total acceptance probability decreases.

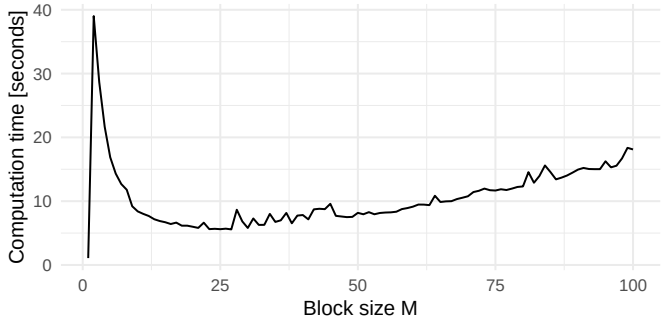
From subplot (b) we see that the computational time is a more complicated. The computation time for sampling without blocking ($M = 1$) is lower than using the sampler with blocking for values of $M = 2, \cdots, 5$. The computation time for blocking decreases until it reaches a minimum at around $M = 25$, where it start to increase.

We want to choose the value of M which encapsulates both a good overall acceptance rate and a modest computation time. A rule of thumb when working with MCMC methods is to have acceptance rates in the range 20% to 50%, as this, one could argue, gives a good balance between exploring the sample space and accepting the new proposals. We decided to use the block size $M = 10$ in our further analysis, as the this gives an acceptance rate of about 45% in addition to low computation times.

Incorporating a block step over the $\boldsymbol{\tau}$ parameters might improve the efficiency of our MCMC sampler if the number of effective samples is higher. The sample autocorrelation for $M = 10$ is plotted in Figure 8c. Comparing to the autocorrelation function with single site updates in Figure 4c, we see that the autocorrelations for these examples seem to approach zero faster (maybe except for $\pi(\tau_{201})$), meaning that the samples are less correlated. However, if the block size M is too large, the low acceptance rate means that the elements of $\boldsymbol{\tau}$ are less likely to be updated, so the correlation increases and the effective sample size gets smaller.



(a) Acceptance rates.



(b) Computation time.

Figure 5: Acceptance rates and computation time including pre-computation for $M = 1, 2, \ldots, 100$.

From Figure 7, we observe that the Markov chain appears to have converged (judging by the histogram and the traceplot). Subplot (c) shows that the autocorrelation for σ_u^2 is higher than it is for MCMC with single site updates (see Figure 3c). This could be attributed to the fact that some blocks of $\boldsymbol{\tau}$ may not be updated as often, and since σ_u^2 is sampled conditional on $\boldsymbol{\tau}$, it may be more correlated from one iteration to the next. This is also reflected in the estimated effective sample size as discussed below.

By use of `proc.time()`, we calculated the computation time to be 368 seconds $\approx$ 6 minutes. The acceptance for individual τ_t was, on average, around 0.451 for $M = 10$. Central estimates of $\pi(\tau_1), \pi(\tau_{201}), \pi(\tau_{366})$, and σ_u^2 together with 95 % credible intervals and estimated ESS are given in Table 2. The estimates are calculated with

a burn-in period of 500, which is based on a visual inspection of the traceplots. The table shows that the results are very similar to that of `mcmc.single`, with which it shares the first two significant digits for all entries except for $\pi(\tau_{366})$ and σ_u^2. We note that the effective sample size also increases drastically for $\pi(\tau_{366})$ and some for $\pi(\tau_1)$. However, the estimated ESS decreases slightly for $\pi(\tau_{201})$ and is more than halved for σ_u^2. This is due to the increased autocorrelation in the samples of σ_u^2.

Table 2: Prediction and credible intervals for some selected entries of $\boldsymbol{\pi}$ and σ_u^2, generated by our MCMC-algorithm with blocking ($M = 10$).

Variable	Prediction	2.5% quantile	97.25% quantile	Estimated ESS
$\pi(\tau_1)$	0.1832	0.1325	0.2470	3788
$\pi(\tau_{201})$	0.3325	0.2783	0.3922	3097
$\pi(\tau_{366})$	0.1246	0.08544	0.1756	18382
σ_u^2	0.010641	0.006201	0.01769	525

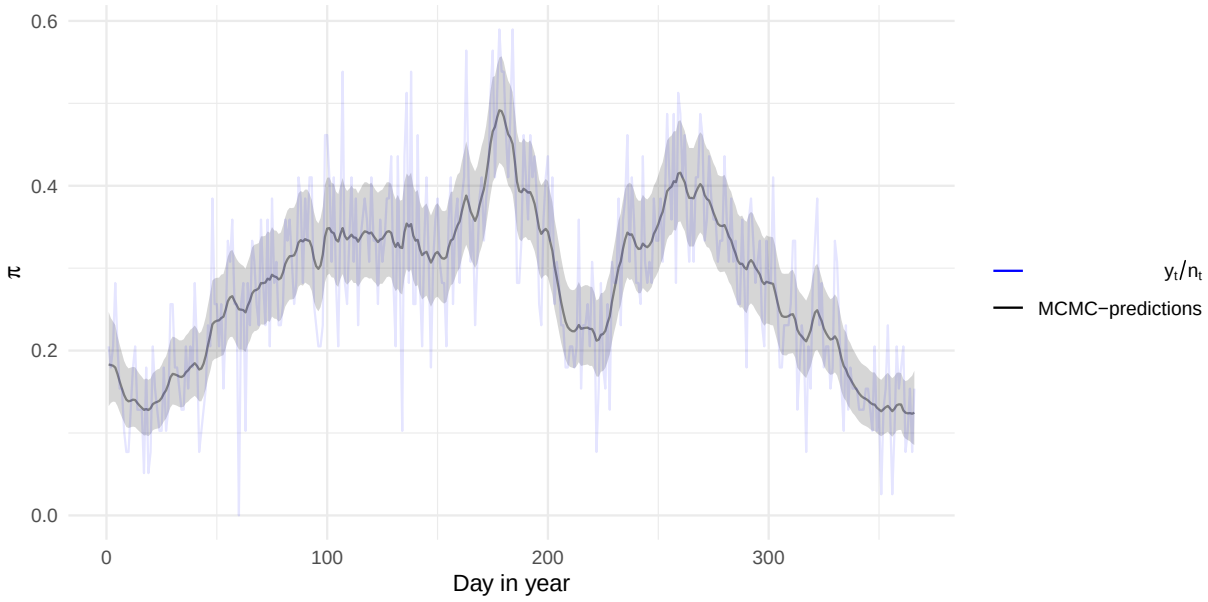
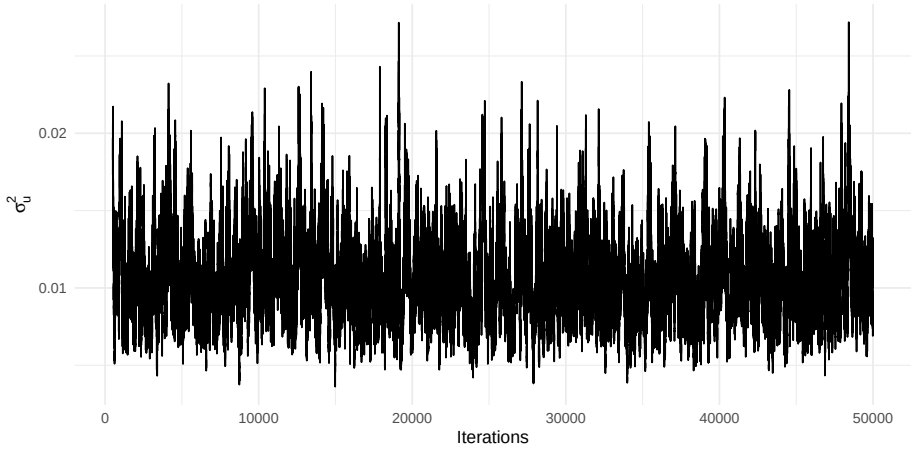
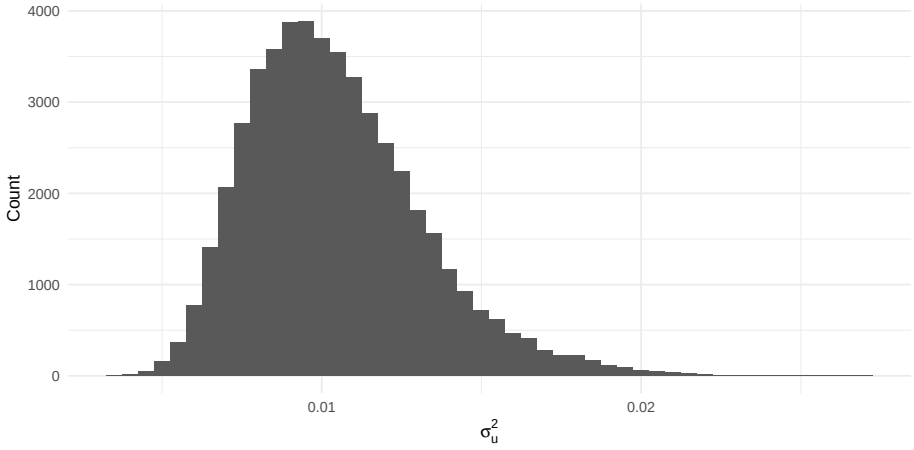


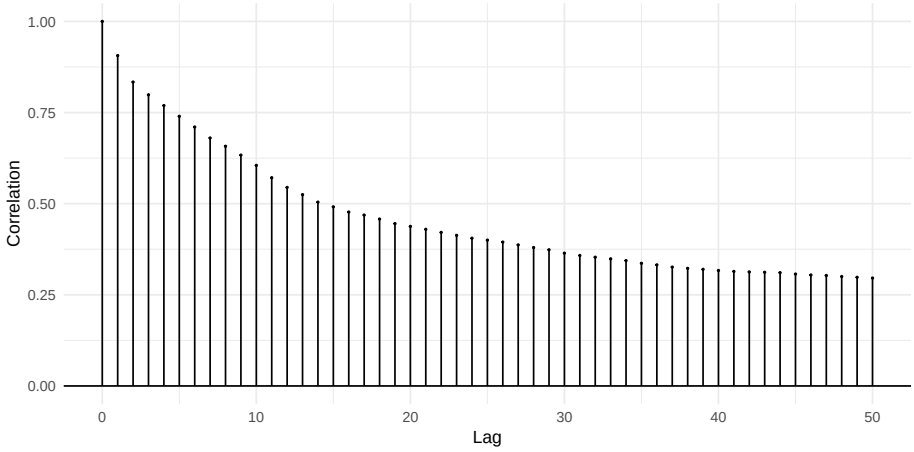
Figure 6: The MCMC-predictions of $\boldsymbol{\pi}$ (black) with associated 95% credible intervals (gray area) using blocking with M = 10. The fractions y_t/n_t are displayed in blue.



(a) Traceplot.

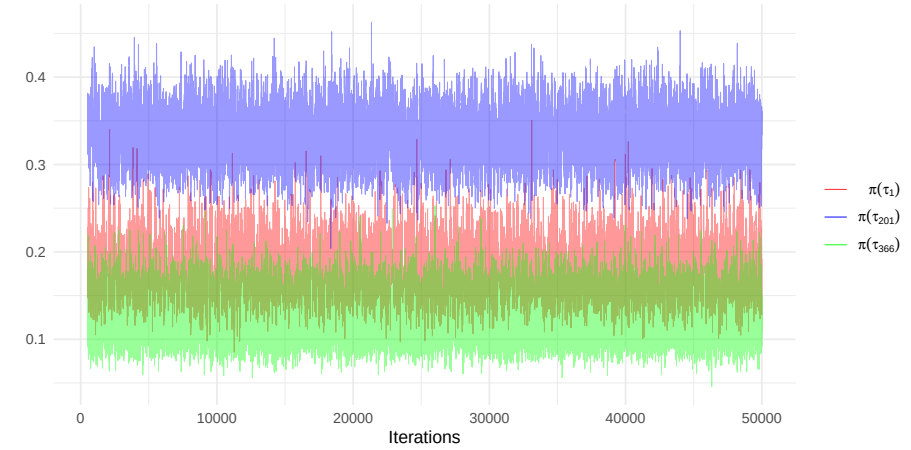


(b) Histogram.

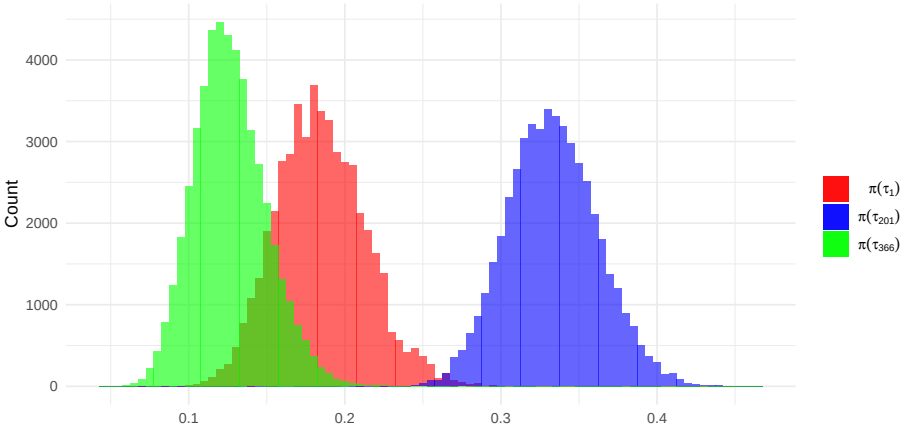


(c) Autocorrelation.

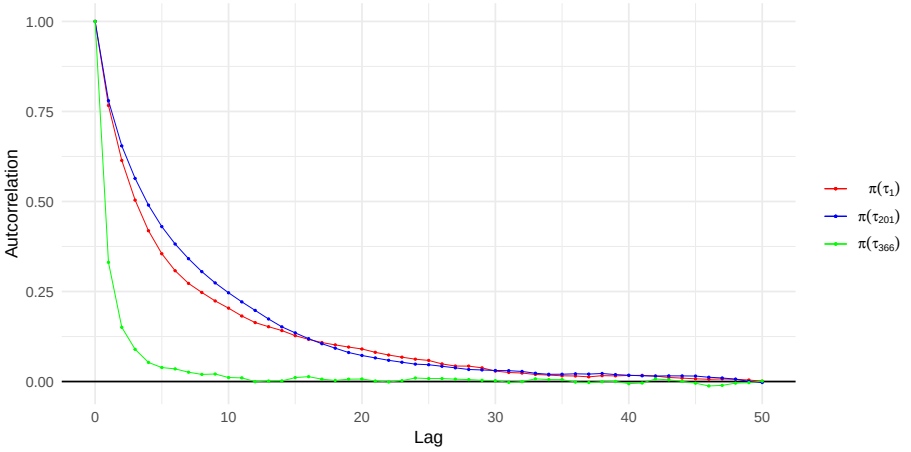
Figure 7: Traceplot (a), histogram (b) and autocorrelation (c) for σ_u^2 using the MCMC algorithm with block updates of $\boldsymbol{\tau}$ and $M = 10$. The histogram and autocorrelation plots are computed with a burn-in period of 500 samples.



(a) Traceplot of the first 1000 iterations using MCMC with blocking for τ_1 in red, τ_{201} in blue and τ_{366} in green.



(b) Histogram.



(c) Autocorrelation.

Figure 8: Traceplot (a), histogram (b) and sample autocorrelation (c) plots for τ_1 in red, τ_{201} in blue and τ_{366} in green. The histogram and autocorrelation plots are computed with a burn-in period of 500 samples.

Problem 2

In this problem, we will use INLA rather than MCMC and compare the results. We start by fitting the same model as in Problem 1 to the Tokyo rainfall data set. The required code is given below.

```
control.inla = list(strategy="simplified.laplace", int.strategy="ccd")
ptm <- proc.time() # To calculate computation time
mod <- inla(n.rain ~ -1 + f(day, model="rw1",
                            hyper = list(theta = list(prior="loggamma", param=c(alpha, beta))),
                            constr=FALSE),
            data=rain, Ntrials=n.years, control.compute=list(config = TRUE),
            family="binomial", verbose=TRUE, control.inla=control.inla)
(proc.time() - ptm)[3] # Computation time
```

a)

Notice that we calculate the computation time above by use of `proc.time()`, which on our computer was 4.068 seconds. Furthermore, we require no prior for the intercept in the model, since there is none. We do, on the other hand, specify a prior for the precision hyper parameter, which corresponds to the log-prior of $\frac{1}{\sigma_u^2}$ from Problem 1. Since $\sigma_u^2 \sim \text{Inv-Gamma}(\alpha, \beta)$, where α and β are shape and scale parameters, respectively, we know that

$$\frac{1}{\sigma_u^2} \sim \text{Gamma}(\alpha, \beta),$$

where β is the rate parameter from the perspective of the gamma distribution (see e.g. here). Consequently, the log-prior is set to be a log-gamma distribution with shape α and scale β.

The predictions of $\boldsymbol{\pi}$ which the above INLA model produces are displayed in Figure 9. The code used to create the plot is given below.

```
# Create plot of y/n and INLA-predictions and save
pi.df <- data.frame(pi = sigm(mod$summary.random$day$mean),
                    lower = sigm(mod$summary.random$day$`0.025quant`),
                    upper = sigm(mod$summary.random$day$`0.975quant`),
                    p = y/n, x = 1:T)

pi.preds <- ggplot(pi.df, aes(x = x)) + geom_line(aes(y = pi, color = "pi")) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
  geom_line(aes(y = p, color = "yt/nt"), alpha = 0.1) +
  xlab("Day in year, t") + ylab(expression(pi)) +
  scale_color_manual(name = " ", values = c("yt/nt" = "blue", "INLA-predictions" = "black"),
                     labels = c(expression(y[t]/n[t]),"INLA-predictions")) + theme_minimal()

ggsave("./figures/pi_preds_inla.pdf", plot = pi.preds, height = 4.0, width = 8.0)
```

Comparing the predictions and corresponding uncertainties generated by INLA to the ones generated by the use of MCMC methods in Figures 2 and 6, we observe that the results are very similar. The INLA-model's predictions of $\pi(\tau_1)$, $\pi(\tau_{201})$, $\pi(\tau_{366})$ and σ_u^2 are listed in Table 3, along with 95% credible intervals. Comparing with Table 1, we see that the predictions of $\pi(\tau_t)$ are very similar, and all entries share the same first two significant digits, including the bounds on the credible intervals. The predictions of σ_u^2 are more different, though still fairly similar, and the credible interval is narrower in INLA-model. This may be due to the fact that the MCMC-sampler exhibits quite a bit of autocorrelation in the sampling of σ_u^2, both with single site updates and block updates.

b)

We want to investigate how robust the results of INLA are with respect to the two inputs to `control.inla`. Hence, we have fitted an INLA model for each of the input combinations listed in Table 4, which also contains the

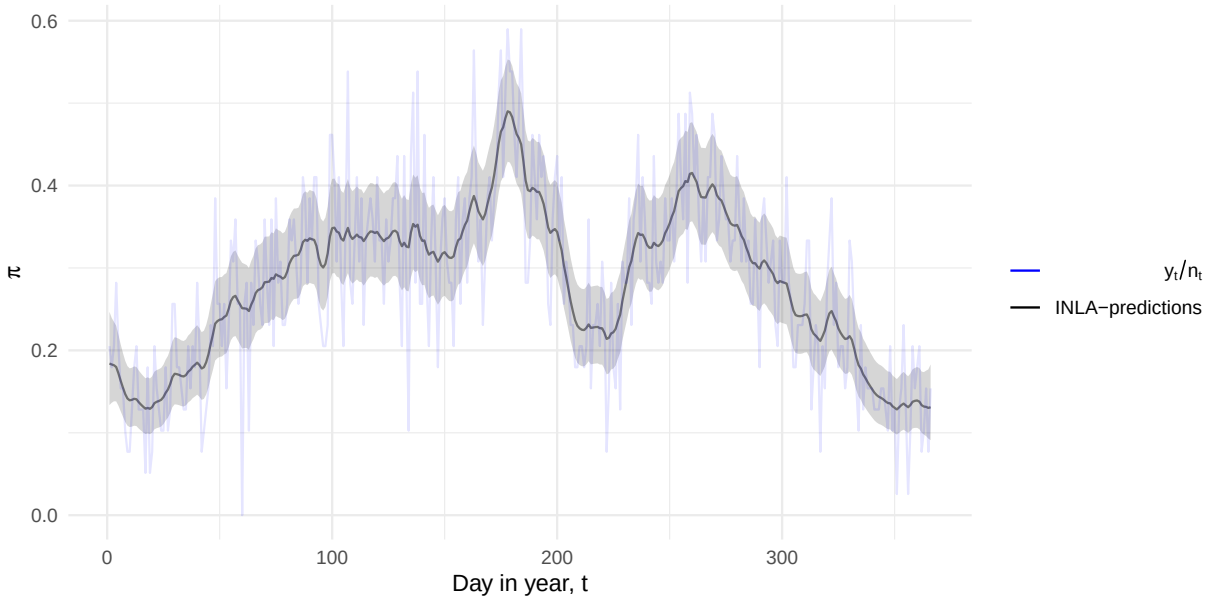


Figure 9: The INLA-predictions of $\boldsymbol{\pi}$ (black) with associated 95% credible intervals (gray area). The proportions y_t/n_t are displayed in blue.

Table 3: Prediction and credible intervals for some selected entries of $\boldsymbol{\pi}$ and σ_u^2, generated by the INLA-model in Problem 2**a)**.

Variable	Prediction	2.5% quantile	97.25% quantile
$\pi(\tau_1)$	0.1840	0.1336	0.2468
$\pi(\tau_{201})$	0.3323	0.2785	0.3911
$\pi(\tau_{366})$	0.1311	0.09100	0.1829
σ_u^2	0.009392	0.009672	0.005916

recorded computation time and the maximum value in the prediction. We notice that there are no large difference in computation times, i.e. they are all in the range 3.5-5 seconds.

Furthermore, we have plotted the predictions which each of the models produces. The result is displayed in Figure 10, and shows that the predictions are mostly the same, especially for $t \leq 100$, where they are practically indistinguishable. However, we observe that using `int.strategy = laplace` yields slightly different predictions, when $t > 100$. This is hard to see from the display, where one can only discern two 'types' of predictions, but from Table 4, we observe that the max values differs for the class of inputs where `int.strategy = laplace`, from which we deduce that this must be the culprit. We conclude that the results are fairly robust to the inputs of`control.inla`, with the exception `int.strategy = laplace`. The code used in this exercise is given below.

```
# Plot prediction for 12 different control input combinations
strategy = c('gaussian', 'simplified.laplace', 'laplace', 'adaptive')
int.strategy = c('ccd', 'grid', 'eb')
controls <- expand.grid(strategy, int.strategy)

pal <- unname(tol()) # Color palette
times <- rep(NA, 12) # Computation times
sc <- list() # For ggplot
p <- ggplot() # Init. plot
max.vec <- rep(0, 12) # Vector for maximal value in preds
combs <- 1:12 # Combination index
for(i in combs){
  # Set inputs
  control.inla = list(strategy=paste(controls[i,1]),
```

	strategy	int.strategy	Computation time	Max value
1	gaussian	ccd	4.1820	0.4902
2	simplified.laplace	ccd	3.7600	0.4901
3	laplace	ccd	4.3310	0.4403
4	adaptive	ccd	3.5140	0.4902
5	gaussian	grid	3.8270	0.4904
6	simplified.laplace	grid	4.3580	0.4903
7	laplace	grid	4.4850	0.4406
8	adaptive	grid	3.6840	0.4904
9	gaussian	eb	3.7530	0.4901
10	simplified.laplace	eb	3.6910	0.4900
11	laplace	eb	3.9420	0.4402
12	adaptive	eb	3.7190	0.4901

Table 4: The different input combinations to `control.inla` and their associated computation time, as well as the maximum value in the prediction.

```
                    int.strategy=paste(controls[i,2]))
  # Fit model and time it
  ptm <- proc.time()
  mod <- inla(n.rain ~ -1 + f(day, model="rw1",
                              hyper = list(theta = list(prior="loggamma", param=c(alpha, beta))),
                              constr=FALSE),
              data=rain, Ntrials=n.years, control.compute=list(config = TRUE),
              family="binomial", verbose=TRUE, control.inla=control.inla)
  times[i] <- (proc.time() - ptm)[3]

  preds <- sigm(mod$summary.random$day$mean)
  max.vec[i] <- max(preds)
  p <- p + geom_line(data = data.frame(x = 1:T, y = preds),
                     aes(x, y, color = paste(i)), size = 0.3)
  sc[[i]] = pal[i]
}

# Plotting
names(sc) <- combs
plot.control <- p + scale_color_manual(name = " ",  values = unlist(sc))+
  xlab("Day in year, t")  + ylab(expression(pi)) + theme_minimal()
ggsave("./figures/inla_control.pdf", plot = plot.control, height = 4.0, width = 8.0)


# Create dataframe of information
summary.df <- data.frame(strategy = controls[, 1],
                   int.strategy = controls[ ,2],
                   time = times, max = max.vec)

# Create table in LaTeX
xtable(comp.times)
```

Note: In hindsight, we have been informed by the lecturer in this course that the difference in predictions is caused by a bug. The predictions are supposed to be completely robust to the inputs of `control.inla` for the present case.

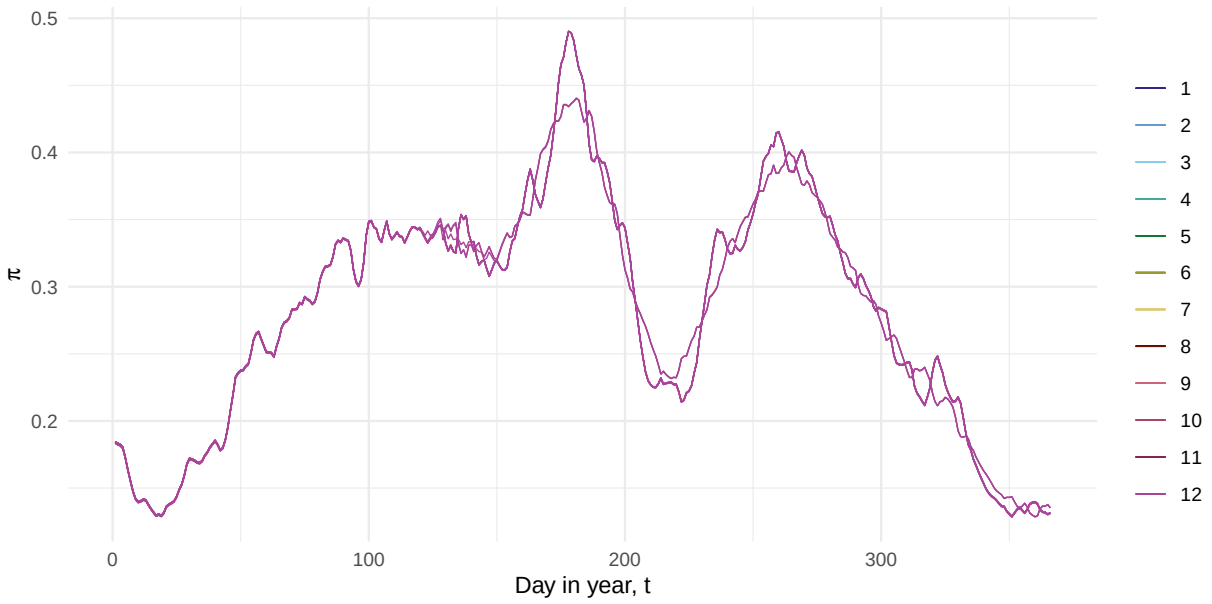


Figure 10: Predictions of the different INLA-models arising from the inputs to `control.inla` listed in Table 4.

c)

Finally, we consider an alternative model formulation to that in (1), where we include an intercept, β, i.e., the model has the linear predictor $\eta_t = \beta + \tau_t$, so we write the model as

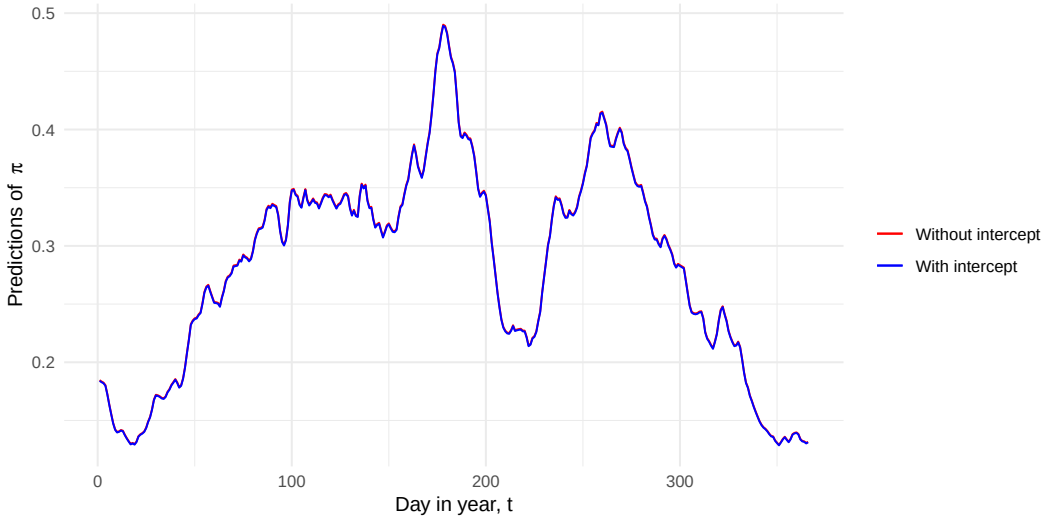
$$y_t|\tau_t, \beta \sim \text{Bin}(n_t, \pi(\eta_t)), \quad \pi(\eta_t) = \frac{\exp(\eta_t)}{1+\exp(\eta_t)}.$$

Additionally, we impose the contraint

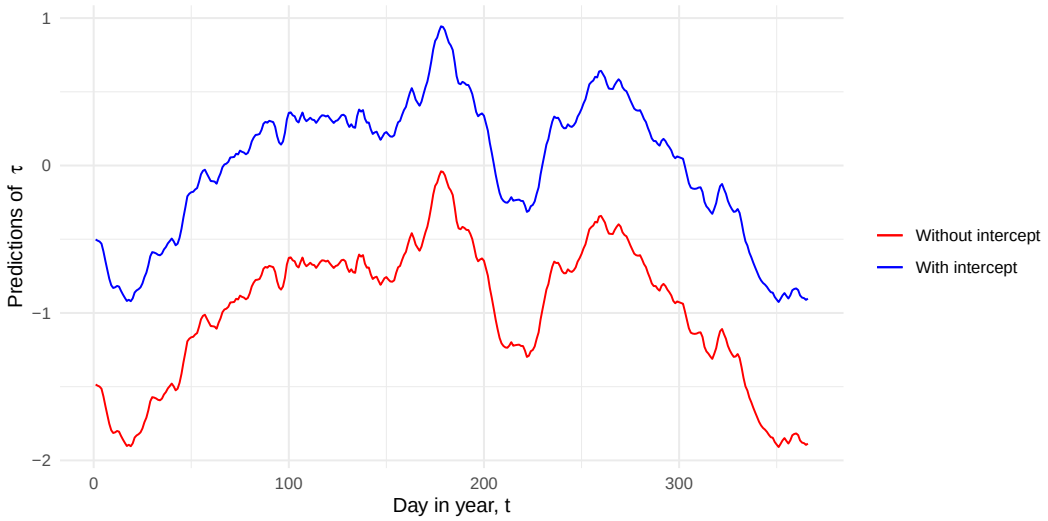
$$\sum_{t=1}^{T} \tau_t = 0, \tag{11}$$

which is done by setting `constr = TRUE`. To compare the two candidate models, we plot their predictions of $\boldsymbol{\pi}$ jointly. The result is displayed in Figure 11a and the corresponding code is given directly below.

```
# Model with intercept and constraint
mod.intercept <- inla(n.rain ~ f(day, model="rw1",
                        hyper = list(theta = list(prior="loggamma", param=c(alpha, beta))),
                        constr=TRUE),
            data=rain, Ntrials=n.years, control.compute=list(config = TRUE),
            family="binomial", verbose=TRUE, control.inla=control.inla)

# Plot predictions together
df <- data.frame(x = 1:T, pred.a =  sigm(mod$summary.random$day$mean),
                 pred.int =  sigm(mod.intercept$summary.random$day$mean +
                 mod.intercept$summary.fixed$mean))
plot.together <- ggplot(df, aes(x = x)) +
geom_line(aes(y = pred.a, color = "Without intercept")) +
  geom_line(aes(y = pred.int, color = "With intercept")) +
  scale_color_manual(name = " ",
  values = c("Without intercept" = "red", "With intercept" = "blue")) +
  xlab("Day in year, t") + ylab(expression("Predictions of "~pi)) + theme_minimal()
ggsave("./figures/inla_together.pdf", plot = plot.together, height = 4.0, width = 8.0)
```



(a)



(b)

Figure 11: (a) Predictions of $\boldsymbol{\tau}$ and (b) predictions of $\boldsymbol{\pi}$ of the model fitted in **2a)** (red) and the model with and intercept and a constraint (blue).

The display reveals that the predictions of the two models are practically identical, and to explain this, we plot the estimated values of $\boldsymbol{\tau}$ for each of the models, as is shown in Figure 11b. Below is the required code.

```
# Plot values of tau
df <- data.frame(x = 1:T, pred.a =  mod$summary.random$day$mean,
                 pred.int =  mod.intercept$summary.random$day$mean)
plot.tau <- ggplot(df, aes(x = x)) + geom_line(aes(y = pred.a, color = "Without intercept")) +
  geom_line(aes(y = pred.int, color = "With intercept")) +
  scale_color_manual(name = " ",
  values = c("Without intercept" = "red", "With intercept" = "blue")) +
  xlab("Day in year, t") + ylab(expression("Predictions of "~tau)) + theme_minimal()
ggsave("./figures/inla_tau.pdf", plot = plot.tau, height = 4.0, width = 8.0)
```

The display clearly illustrates that even though the model is constrained by (11), the values of $\boldsymbol{\tau}$ can fit to exactly the same 'shape' and then be shifted down by the intercept, β, yielding identical predictions for $\boldsymbol{\pi}$. In a sense, the two models are just as 'flexible' and therefore produce equivalent predictions.