TMA4300; Exercise 2

Martinius Singdahlsen, Ola Rasmussen, Johan Lagardére

Contents

Problem A

Introduction

This problem is about the Tokyo rainfall from 1951 to 1989. We will consider the response to be the number of days, for each date, in this time period where the amount of rainfall exceeded 1mm:

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

$$\begin{aligned}\pi(\tau_t)) &= \frac{e^{\tau_t}}{1+e^{\tau_t}} \\ &= \frac{1}{1+e^{-\tau_t}},\end{aligned}$$

where $n_t = 10$ when $t = 60$ (February 29th), and $n_t = 39$ for all the other days. $\pi(\tau_t)$ is the probability of the rainfall exceeding 1mm for days $t = 1, \ldots, T$ where $T = 366$. We assume conditional independence among the $y_t|\tau_t$ for all $t = 1, \ldots, 366$.

a)

In this part of the problem we will start with plotting the rain data, and then comment any pattern that might appear.

```
# Loading the rainfall data
load(file = "rain.rda")
# Plotting response as a function of time
plot(n.rain ~ day, rain, type = "l", lwd = 2, xlab = "t (day)",
    ylab = "Amount of rain exceeding 1mm")
# Adding February 29th
points(60, 0, col = "blue", pch = 19)
```

We see here in Figure 1 that the amount of rainwater exceeding 1mm increases in the period $t = 1, .., 180$ (Start of January to end of June), decreases around $t = 200$ (Start of July to mid august), increases again in the period $t = 220, \ldots, 260$ (Mid August to mid September), and then decreases again in the period $t = 260, \ldots, 366$ (Mid September to the end of the year).

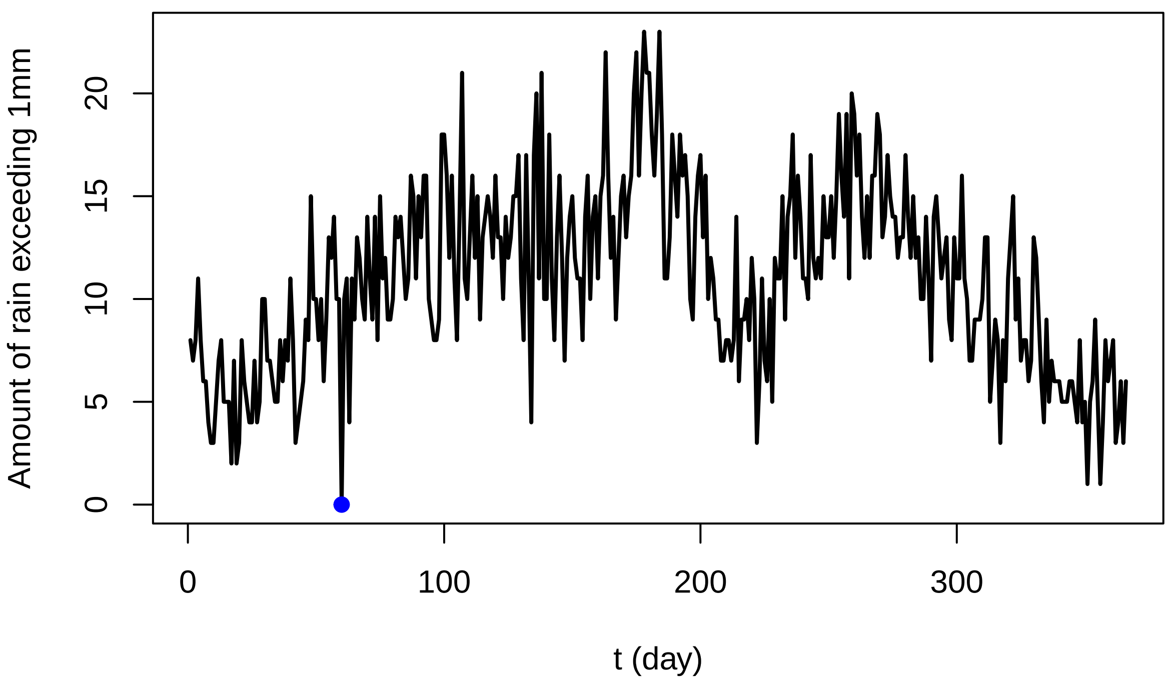


Figure 1: Plotted response as a function of t.

b)

In this part of the problem we will obtain the likelihood of $y_t|\pi(\tau_t)$ for $t = 1, \dots, 366$. To do this we will use the following formula for the likelihood,

$$L_x(\boldsymbol{\theta}) = \prod_{t=1}^{T} f(x_t|\boldsymbol{\theta}). \tag{1.1}$$

In our case $x = y$, $x_t = y_t$ and $\boldsymbol{\theta} = \pi(\tau_t)$. We know that $y_t|\tau_t = y_t|\pi(\tau_t) \sim Bin(n_t, \pi(\tau_t))$ are independent. We then get that the likelihood is

$$\begin{aligned} L_{\boldsymbol{y}}(\pi(\tau_t)) &= \prod_{t=1}^{T} f(y_t|\pi(\tau_t)) \\ &= \prod_{t=1}^{T} \binom{n_t}{y_t} \pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{n_t-y_t} \\ &\propto \prod_{t=1}^{T} \pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{n_t-y_t}. \end{aligned}$$

So we get,

$$L_{\boldsymbol{y}}(\pi(\tau_t)) \propto \prod_{t=1}^{T} \pi(\tau_t)^{y_t}(1-\pi(\tau_t))^{n_t-y_t}. \tag{1.2}$$

c)

Now we are interested in the conditional $p(\sigma_u^2|\boldsymbol{y},\boldsymbol{\tau})$.

We get that,

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

We already have

$$p(\boldsymbol{\tau}|\sigma_u^2) = \prod_{t=2}^{T} \frac{1}{\sigma_u} e^{-\frac{1}{2\sigma_u^2}(\tau_t-\tau_{t-1})^2} \tag{1.3}$$

$$p(\sigma_u^2) = \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} e^{-\frac{\beta}{\sigma_u^2}}, \tag{1.4}$$

where $\tau_t = \tau_{t-1} + u_t$, $u_t \overset{\text{iid}}{\sim} N(0,\sigma_u^2)$, $\boldsymbol{y} = (y_1,\ldots,y_T)^\top$, $\boldsymbol{\tau} = (\tau_1,\ldots,\tau_T)^\top$ and $\boldsymbol{\pi} = (\pi(\tau_1),\ldots,\pi(\tau_T))^\top$.

When we condition on $\boldsymbol{y}$ and $\boldsymbol{\tau}$, $p(\boldsymbol{y}|\boldsymbol{\tau}) = L_{\boldsymbol{y}}(\pi(\tau_t))$, as seen above, becomes a scalar. So it can be ignored. We then have,

$$\begin{aligned} p(\sigma_u^2|\boldsymbol{y},\boldsymbol{\tau}) &\propto p(\boldsymbol{\tau}|\sigma_u^2)\cdot p(\sigma_u^2) \\ &= \left\{\prod_{t=2}^{T}\frac{1}{\sigma_u}e^{-\frac{1}{2\sigma_u^2}(\tau_t-\tau_{t-1})^2}\right\} \times \left\{\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} e^{-\frac{\beta}{\sigma_u^2}}\right\} \\ &\propto \left\{\frac{1}{\sigma_u^{T-1}}e^{\sum_{n=2}^{T}\left(-\frac{1}{2\sigma_u^2}(\tau_t-\tau_{t-1})^2\right)}\right\} \times \left\{\left(\frac{1}{\sigma_u^2}\right)^{\alpha+1} e^{-\frac{\beta}{\sigma_u^2}}\right\} \\ &= \left(\frac{1}{\sigma_u^2}\right)^{\left(\alpha+\frac{T-1}{2}\right)+1} e^{-\frac{\beta}{\sigma_u^2}+\sum_{n=2}^{T}\left(-\frac{1}{2\sigma_u^2}(\tau_t-\tau_{t-1})^2\right)} \\ &= \left(\frac{1}{\sigma_u^2}\right)^{\left(\alpha+\frac{T-1}{2}\right)+1} e^{-\frac{1}{\sigma_u^2}\left(\beta+\frac{1}{2}\sum_{n=2}^{T}(\tau_t-\tau_{t-1})^2\right)}. \end{aligned}$$

This is an inverse gamma distribution with shape parameter $\left(\alpha+\frac{T-1}{2}\right)$ and scale parameter $\left(\beta+\frac{1}{2}\sum_{n=2}^{T}(\tau_t-\tau_{t-1})^2\right)$, so

$$\sigma_u^2|\boldsymbol{y},\boldsymbol{\tau} \sim IG\left(\alpha+\frac{T-1}{2},\ \ \beta+\frac{1}{2}\sum_{t=2}^{T}(\tau_t-\tau_{t-1})^2\right). \tag{1.5}$$

d)

Letting the conditional prior proposal distribution $Q(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = p(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2)$, we will show that the resulting acceptance probability, $\alpha(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})$, is given by the ratio of likelihoods:

$$\alpha(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \min\left\{1, \frac{p(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}'_{\mathcal{I}})}{p(\boldsymbol{y}_{\mathcal{I}}|\boldsymbol{\tau}_{\mathcal{I}})}\right\}. \tag{1.6}$$

The formula for the acceptance probability is $\alpha(\mathsf{y}|\mathsf{x}) = \min\left\{1, \frac{\pi(\mathsf{y})}{\pi(\mathsf{x})}\frac{Q(\mathsf{x}|\mathsf{y})}{Q(\mathsf{y}|\mathsf{x})}\right\}$. In our case, the acceptance probability is,

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

Writing out $p(\boldsymbol{\tau}'_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})$ we get,

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

and the same for $p(\boldsymbol{\tau}_{\mathcal{I}}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y})$,

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

So then we get that,

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

e)

Now we will implement an MCMC sampler for the posterior $p(\boldsymbol{\pi}, \sigma_u^2|\boldsymbol{y})$. We will use the Gibbs step for sampling from Equation 1.5, with $\alpha = 2$ and $\beta = 0.05$. That means that we will accept every single sample.

We will use the Metropolis-Hastings step for the conditional prior $p(\tau_t|\boldsymbol{\tau}_{-t}, \sigma_u)$. That means that we will accept new taus according to the acceptance probability in Equation 1.6. They will be sampled from the conditional normal distribution with mean,

$$\boldsymbol{\mu}_{A|B} = -\boldsymbol{Q}_{AA}^{-1}\boldsymbol{Q}_{AB}\boldsymbol{x}_B, \tag{1.7}$$

and precision,

$$\boldsymbol{Q}_{A|B} = \boldsymbol{Q}_{AA}, \tag{1.8}$$

where $Q = Prec(\boldsymbol{\tau}|\sigma_u^2)$. Now we will find the acceptance probability on the log scale.

$$\begin{aligned}
log\left(\frac{p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}'_\mathcal{I})}{p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}_\mathcal{I})}\right) &= log(p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}'_\mathcal{I})) - log(p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}_\mathcal{I})) \\
&= log\Bigg(\pi(\tau'_\mathcal{I})^{y_\mathcal{I}}(1-\pi(\tau'_\mathcal{I}))^{n_\mathcal{I}-y_\mathcal{I}}\Bigg) - log\Bigg(\pi(\tau_\mathcal{I})^{y_\mathcal{I}}(1-\pi(\tau_\mathcal{I}))^{n_\mathcal{I}-y_\mathcal{I}}\Bigg) \\
&= \Bigg(y_\mathcal{I}\cdot log\bigg(\frac{1}{1+e^{-\tau'_\mathcal{I}}}\bigg) + (n_\mathcal{I}-y_\mathcal{I})\cdot log\bigg(\frac{e^{-\tau'_\mathcal{I}}}{1+e^{-\tau'_\mathcal{I}}}\bigg)\Bigg) \\
&- \Bigg(y_\mathcal{I}\cdot log\bigg(\frac{1}{1+e^{-\tau_\mathcal{I}}}\bigg) + (n_\mathcal{I}-y_\mathcal{I})\cdot log\bigg(\frac{e^{-\tau_\mathcal{I}}}{1+e^{-\tau_\mathcal{I}}}\bigg)\Bigg) \\
&= \Bigg(-y_\mathcal{I}\cdot log(1+e^{-\tau'_\mathcal{I}}) + (n_\mathcal{I}-y_\mathcal{I})\cdot\bigg((-\tau'_\mathcal{I}) - log(1+e^{-\tau'_\mathcal{I}})\bigg)\Bigg) \\
&- \Bigg(-y_\mathcal{I}\cdot log(1+e^{-\tau_\mathcal{I}}) + (n_\mathcal{I}-y_\mathcal{I})\cdot\bigg((-\tau_\mathcal{I}) - log(1+e^{-\tau_\mathcal{I}})\bigg)\Bigg) \\
&= y_\mathcal{I}\Bigg(log(1+e^{-\tau_\mathcal{I}}) - log(1+e^{-\tau'_\mathcal{I}})\Bigg) \\
&+ (n_\mathcal{I}-y_\mathcal{I})\Bigg((\tau_\mathcal{I}-\tau'_\mathcal{I}) + \bigg(log(1+e^{-\tau_\mathcal{I}}) - log(1+e^{-\tau'_\mathcal{I}})\bigg)\Bigg).
\end{aligned}$$

Generally, the acceptance probability then becomes,

$$\alpha(\boldsymbol{\tau}'_\mathcal{I}|\boldsymbol{\tau}_{-\mathcal{I}}, \sigma_u^2, \boldsymbol{y}) = \min\Bigg\{0, exp\Bigg\{\sum_{\mathcal{I}=1}^{T} log\left(\frac{p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}'_\mathcal{I})}{p(\boldsymbol{y}_\mathcal{I}|\boldsymbol{\tau}_\mathcal{I})}\right)\Bigg\}\Bigg\}, \tag{1.9}$$

where $\mathcal{I} \in [1, T]$.

```r
# Acceptance probability from Eq. 1.9
acceptProb <- function(y, tau, tauProp, n) {
    firstTerm <- y * (log(1 + exp(-tau)) - log(1 + exp(-tauProp)))
    secondTerm <- (n - y) * ((tau - tauProp) + (log(1 + exp(-tau)) -
        log(1 + exp(-tauProp))))
    return(min(1, exp(sum(firstTerm + secondTerm))))
}
# MCMC algorithm
MC <- function(rain, tau. = 1, M = 50000, seed = 98, burnin = 1000,
    printEvery = 100, verbose = F) {
    set.seed(seed)
    TT <- 366
    tauMat <- matrix(0, nrow = M + 1, ncol = TT)
    tauMat[1, ] <- tau.
    sigma2Vec <- numeric(M)
    tau <- tauMat[1, ]
    accepts <- matrix(F, nrow = M, ncol = TT)
    QQ <- tridiag(c(1, rep(2, TT - 2), 1), rep(-1, TT - 1), rep(-1,
        TT - 1))
    startTime <- proc.time()[3]
    for (i in 1:M) {
        if (((i%%printEvery) == 0) && verbose) {
            # print current state and expected computation
            # time left:
            currState <- paste(i, collapse = ", ")
            print(paste0("current state for iteration ", i, "/",
                M, ": ", currState))
            currTime <- proc.time()[3]
            timeTaken <- currTime - startTime
            fracDone <- (i - 1)/M
            fracLeft <- 1 - fracDone
            timeLeftEst <- (timeTaken/fracDone) * fracLeft
            print(paste0("estimated time left: ", round(timeLeftEst),
                " seconds"))
        }
        # Drawing sigma2 from Eq. 1.6
        sigma2 <- rinvgamma(1, shape = (2 + (TT - 1)/2), rate = (0.05 +
            0.5 * sum(diff(tau)^2)))
        # Defining precision matrix
        Q <- QQ/sigma2
        A <- 1
        B <- -1
        QAA <- Q[A, A]
        QAB <- Q[A, B]
```

```
    # Defining mean from Eq. 1.7
    muAcondB <- -QAA^(-1) * QAB[1] * tau[B][1]
    # Defining Precision from Eq. 1.8
    QAcondB <- QAA
    # Drawing new tau from normal distribution
    tauProp <- rnorm(1, muAcondB, sqrt(QAcondB^(-1)))
    # Accept/reject step
    if (runif(1) < acceptProb(rain$n.rain[A], tau[A], tauProp,
        rain$n.year[A])) {
        tau[A] <- tauProp
        accepts[i, A] <- T
    }
    for (j in 2:(TT - 1)) {
        A <- j
        B <- -j
        QAA <- Q[A, A]
        QAB <- Q[A, B]
        # Defining mean from Eq. 1.7
        muAcondB <- -(QAA^(-1) * QAB[(j - 1):j]) %*% tau[B][(j -
            1):j]
        # Defining Precision from Eq. 1.8
        QAcondB <- QAA
        # Drawing new tau from normal distribution
        tauProp <- rnorm(1, muAcondB, sqrt(QAcondB^(-1)))
        # Accept/reject step
        if (runif(1) < acceptProb(rain$n.rain[A], tau[A],
            tauProp, rain$n.year[A])) {
            tau[A] <- tauProp
            accepts[i, A] <- T
        }
    }
    A <- TT
    B <- -TT
    QAA <- Q[A, A]
    QAB <- Q[A, B]
    # Defining mean from Eq. 1.7
    muAcondB <- -QAA^(-1) * QAB[TT - 1] * tau[B][TT - 1]
    # Defining Precision from Eq. 1.8
    QAcondB <- QAA
    # Drawing new tau from normal distribution
    tauProp <- rnorm(1, muAcondB, sqrt(QAcondB^(-1)))
    # Accept/reject step
    if (runif(1) < acceptProb(rain$n.rain[A], tau[A], tauProp,
        rain$n.year[A])) {
```

```r
            tau[A] <- tauProp
            accepts[i, A] <- T
        }
        tauMat[i + 1, ] <- tau
        sigma2Vec[i] <- sigma2
    }
    acceptRate <- mean(accepts)
    currTime <- proc.time()[3]
    print(paste0("Total time taken: ", round((currTime - startTime)/60,
        digits = 2), " minutes"))
    print(paste0("Acceptance rate: ", round(acceptRate, digits = 4)))
    return(list(taumatrix = tauMat[(burnin + 1):(M + 1), ], sigma2 = sigma2Vec[(burnin +
        1):M], acceptrate = acceptRate))
}
results <- MC(rain)
```

```
## [1] "Total time taken: 3.68 minutes"
## [1] "Acceptance rate: 0.9171"
```

```r
taus = results$taumatrix
sigma2 <- results$sigma2
acceptRate = results$acceptrate
```

```r
par(mfrow = c(3, 1))
hist(expit(taus[, 1]), freq = F, breaks = 100, xlim = c(0, 0.5))
abline(v = mean(expit(taus[, 1])), lwd = 2, col = "blue")
abline(v = quantile(expit(taus[, 1]), prob = c(0.025, 0.975)),
    lwd = 2, lty = 2, col = "blue")
abline(v = rain$n.rain[1]/rain$n.years[1], lwd = 2, col = "red")
legend("topright", inset = 0.05, legend = c("Mean of expit of tau1",
    "y1/tau1"), lty = 1, col = c("blue", "red"), cex = 0.6)
hist(expit(taus[, 201]), freq = F, breaks = 100, xlim = c(0,
    0.5))
abline(v = mean(expit(taus[, 201])), lwd = 2, col = "blue")
abline(v = quantile(expit(taus[, 201]), prob = c(0.025, 0.975)),
    lwd = 2, lty = 2, col = "blue")
abline(v = rain$n.rain[201]/rain$n.years[201], lwd = 2, col = "red")
legend("topright", inset = 0.05, legend = c("Mean of expit of tau201",
    "y201/tau201"), lty = 1, col = c("blue", "red"), cex = 0.6)
hist(expit(taus[, 366]), freq = F, breaks = 100, xlim = c(0,
    0.5))
abline(v = mean(expit(taus[, 366])), lwd = 2, col = "blue")
abline(v = quantile(expit(taus[, 366]), prob = c(0.025, 0.975)),
```

```
    lwd = 2, lty = 2, col = "blue")
abline(v = rain$n.rain[366]/rain$n.years[366], lwd = 2, col = "red")
legend("topright", inset = 0.05, legend = c("Mean of expit of tau366",
    "y366/tau366"), lty = 1, col = c("blue", "red"), cex = 0.6)
```

```
par(mfrow = c(3, 1))
plot(expit(taus[, 1]), type = "l", main = "Traceplot of expit of tau_1")
plot(expit(taus[, 201]), type = "l", main = "Traceplot of expit of tau_201")
plot(expit(taus[, 366]), type = "l", main = "Traceplot of expit of tau_366")
```

```
par(mfrow = c(3, 1))
acf(expit(taus[, 1]))
acf(expit(taus[, 201]))
acf(expit(taus[, 366]))
```

```
means <- c()
for (i in 1:366) {
    means[i] <- mean(expit(taus[, i]))
}
plot(abs(means - (rain$n.rain/rain$n.years)), type = "l", ylim = c(0,
    1), xlab = "t")
abline(h = mean(abs(means - (rain$n.rain/rain$n.years))), col = "blue")
legend("topright", inset = 0.05, legend = c("error between pi and yt/nt as a function of
    "Mean of the error between pi and yt/nt"), lty = 1, col = c("black",
    "blue"), cex = 0.6)
```

```
par(mfrow = c(3, 1))
plot(sigma2, type = "l", main = "Traceplot of sigma2")
hist(sigma2, freq = F, breaks = 100)
abline(v = mean(sigma2), lwd = 3, col = "blue")
abline(v = quantile(sigma2, probs = c(0.025, 0.975)), lwd = 3,
    lty = 2, col = "blue")
acf(sigma2)
```

We see in Figure 3 that there is currently no evidence that the Markov Chain has not converged. We also see in Figure 5 that there is little to no difference between our MCMC samples and the data we are given.

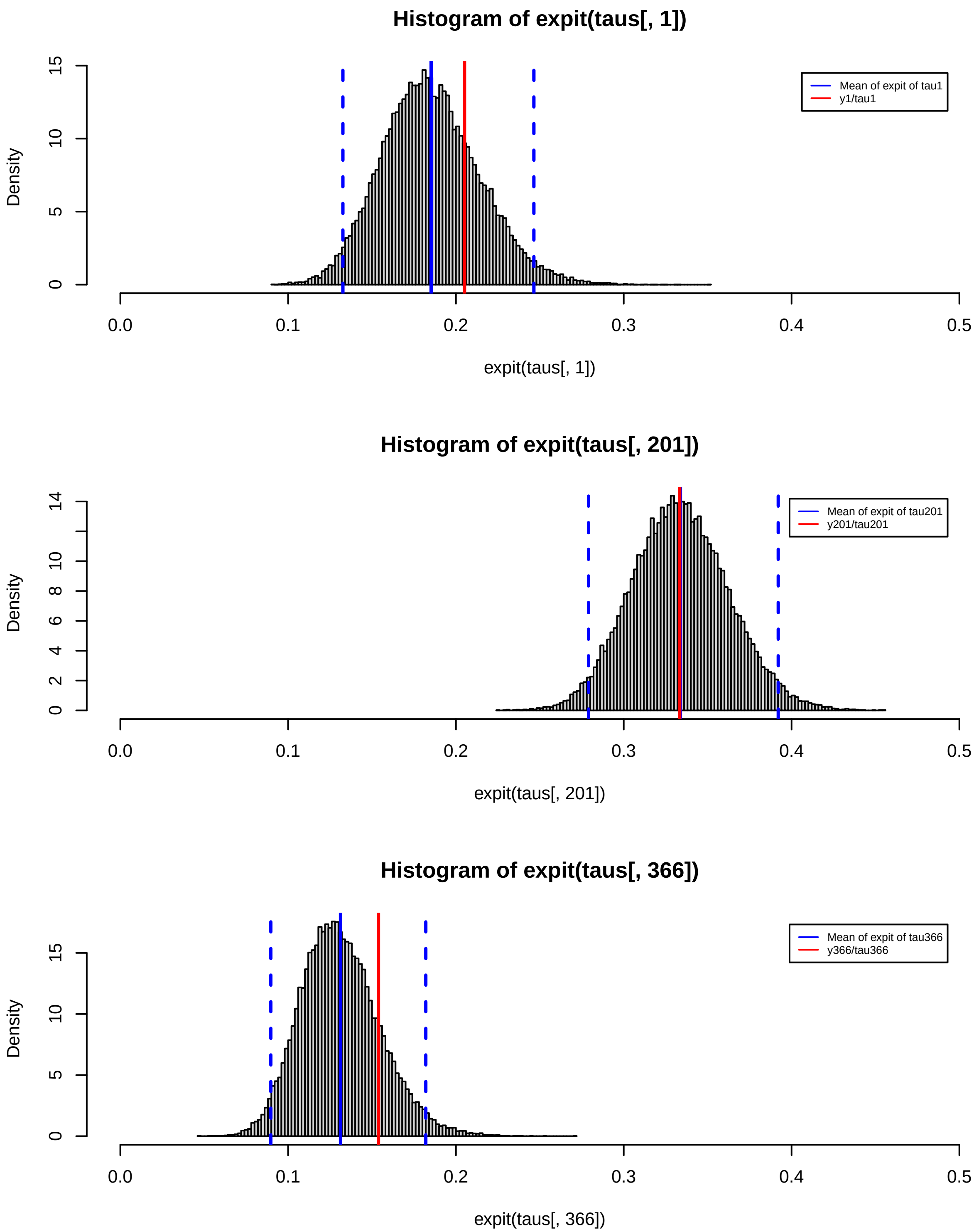


Figure 2: Histograms of expit of tau_1, tau_{201}, and tau_{366}.

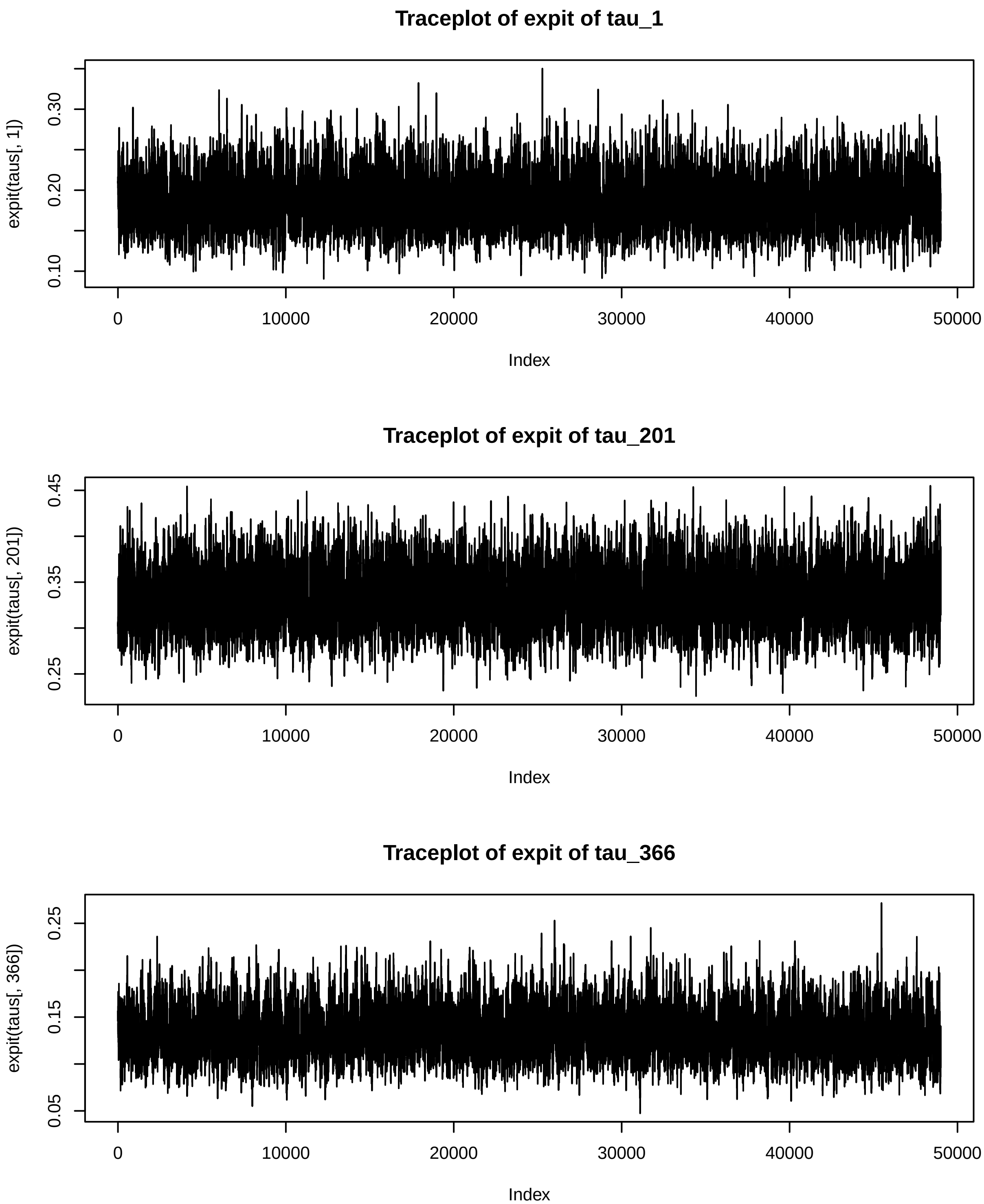


Figure 3: Traceplots of expit of tau_1, tau_{201}, and tau_{366}.

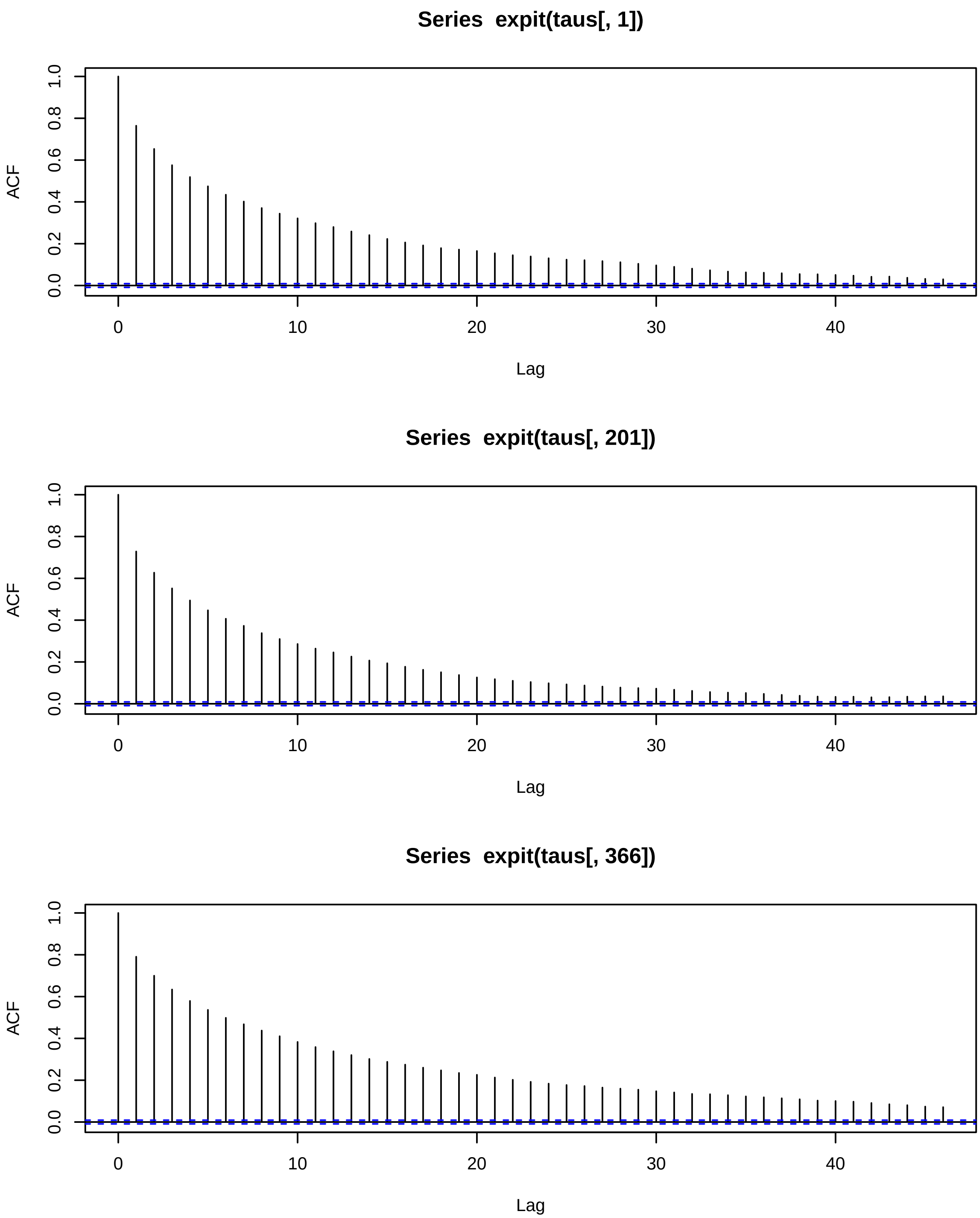


Figure 4: Estimated autocorrelation function for expit of tau_1, tau_{201}, and tau_{366}.

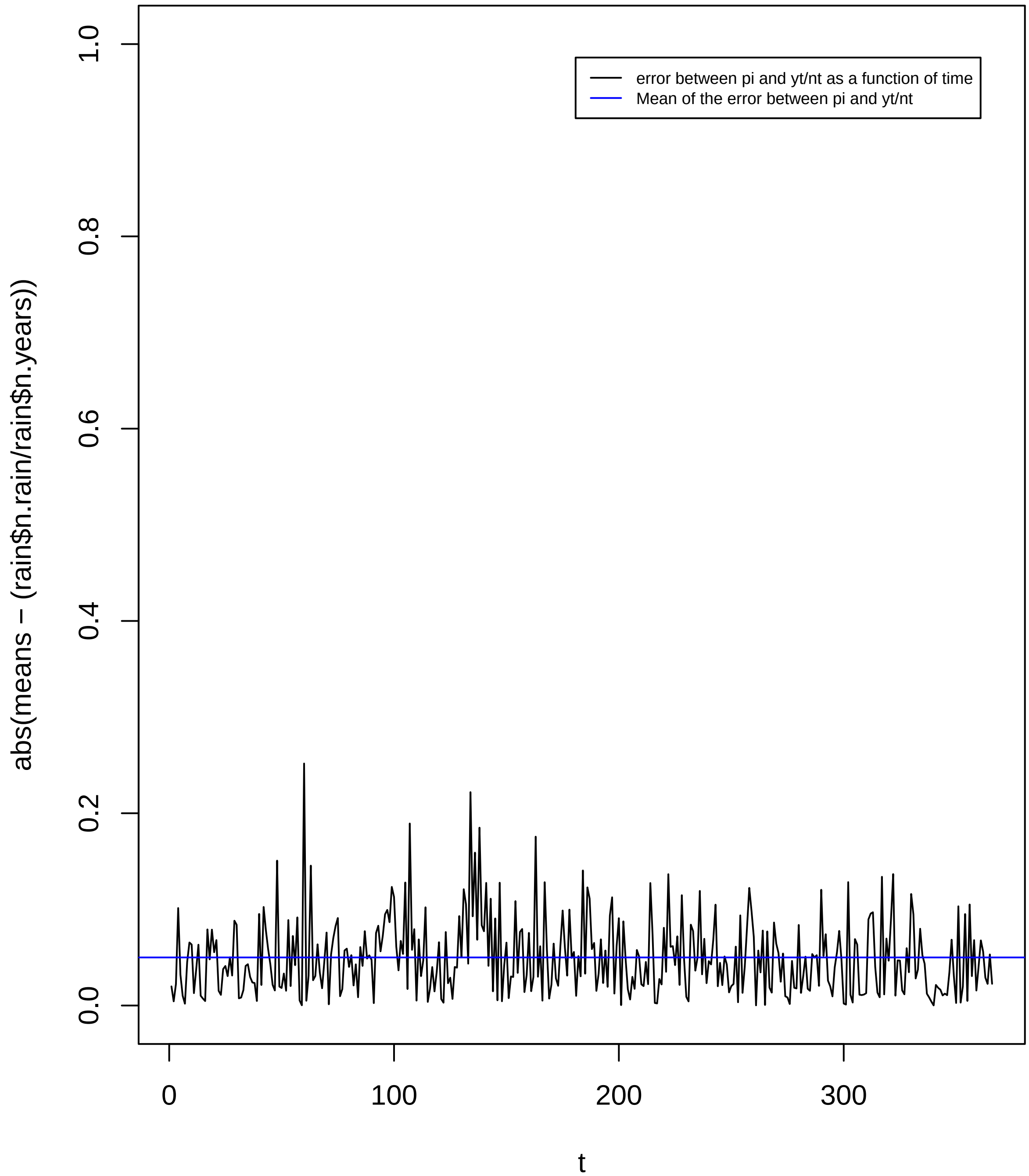


Figure 5: Associated uncertainties of pi and y_t/n_t.

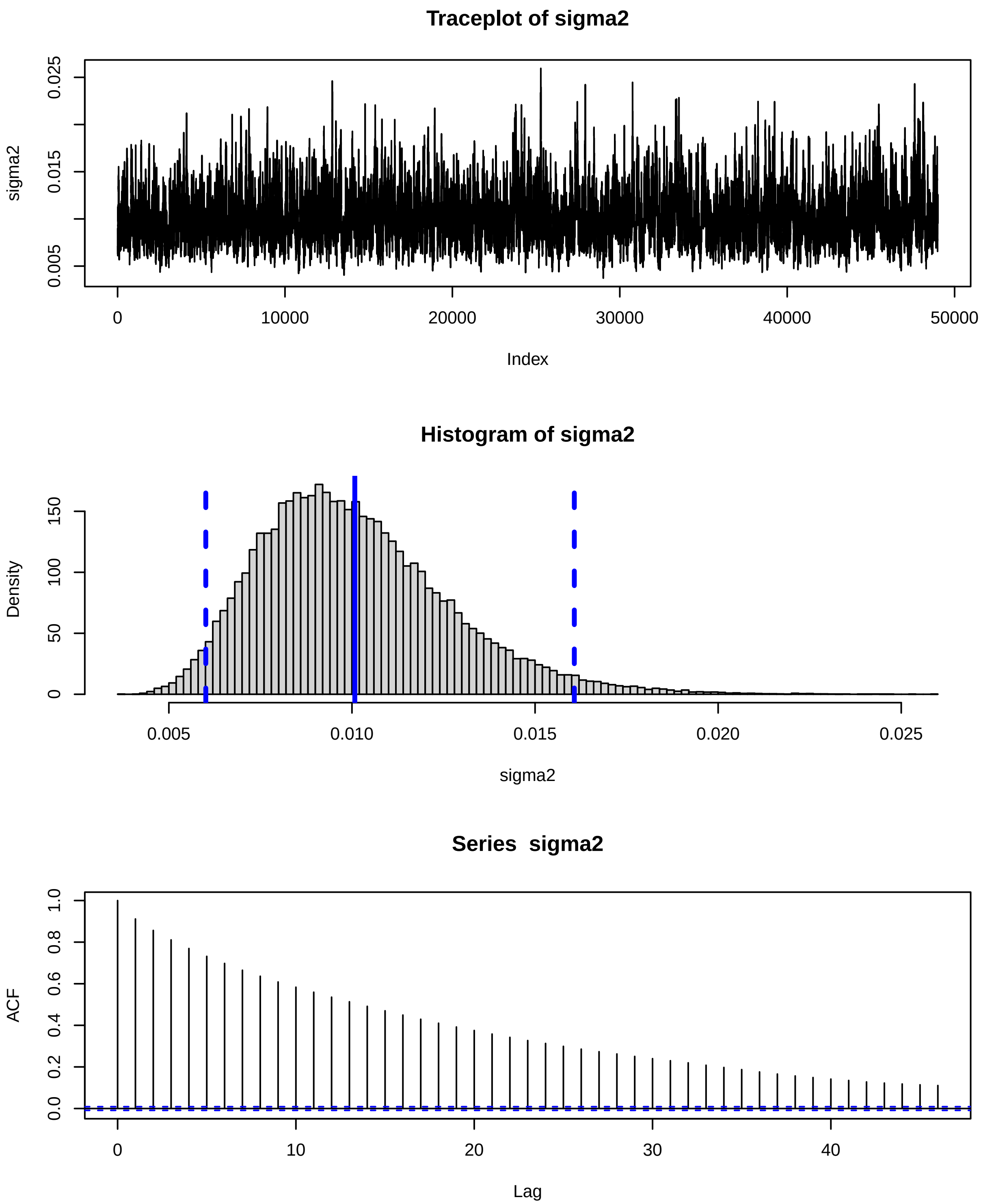


Figure 6: Traceplot, histogram and estimated autocorrelation function for Sigma2.

f)

Here we want to make an MCMC algorithm that uses block updates rather than updating every tau individually. This is done by sampling from a multivariate normal distribution using matrices of dimension N, but the last matrix may be smaller.

```
MCBLOCK <- function(rain, N, tau. = 1, M = 50000, seed = 98,
    burnin = 1000, printEvery = 100, verbose = F) {
    set.seed(seed)
    TT <- 366
    tauMat <- matrix(0, nrow = M + 1, ncol = TT)
    tauMat[1, ] <- tau.
    sigma2Vec <- numeric(M)
    tau <- tauMat[1, ]
    accepts <- matrix(F, nrow = M, ncol = TT)
    QQ <- tridiag(c(1, rep(2, TT - 2), 1), rep(-1, TT - 1), rep(-1,
        TT - 1))
    stepp <- seq(1, TT, N)
    startTime <- proc.time()[3]
    for (i in 1:M) {
        if (((i%%printEvery) == 0) && verbose) {
            # print current state and expected computation
            # time left:
            currState <- paste(i, collapse = ", ")
            print(paste0("current state for iteration ", i, "/",
                M, ": ", currState))
            currTime <- proc.time()[3]
            timeTaken <- currTime - startTime
            fracDone <- (i - 1)/M
            fracLeft <- 1 - fracDone
            timeLeftEst <- (timeTaken/fracDone) * fracLeft
            print(paste0("estimated time left: ", round(timeLeftEst),
                " seconds"))
        }
        # Drawing sigma2 from Eq. 1.6
        sigma2 <- rinvgamma(1, shape = (2 + (TT - 1)/2), rate = (0.05 +
            0.5 * sum(diff(tau)^2)))
        # Defining precision matrix
        Q <- QQ/sigma2
        # Precomputing alle the precision matrices we need
        QAAa1 <- Q[1:N, 1:N]
        QABa1 <- Q[1:N, -(1:N)]
        Qa1 <- solve(QAAa1) %*% QABa1[, 1]
        QAAmid <- Q[2:(N + 1), 2:(N + 1)]
        QABmid <- Q[2:(N + 1), -(2:(N + 1))]
```

```
Qmid <- solve(QAAmid) %*% QABmid[, 1:2]
# a = 1
tauI <- tau[1:N]
# Defining mean from Eq. 1.7
muAcondB <- -Qa1 * tau[N + 1]
# Defining Precision from Eq. 1.8
QAcondB <- QAAa1
# Drawing new taus from multivariate normal
# distribution
tauProp <- t(chol(solve(QAcondB))) %*% rnorm(N) + muAcondB
# Accept/reject step
if (runif(1) < acceptProb(rain$n.rain[1:N], tauI, tauProp,
    rain$n.year[1:N])) {
    tau[1:N] <- tauProp
    accepts[1, 1:N] <- TRUE
}
if (length(stepp) > 2) {
    # a > 1 and b < 366
    for (j in stepp[2:(ceiling(TT/N - 1))]) {
        A = j:(j + N - 1)
        B = -A
        tauI <- tau[A]
        tauminusI <- tau[B]
        # Defining mean from Eq. 1.7
        muAcondB <- -Qmid %*% tauminusI[(j - 1):(j)]
        # Defining Precision from Eq. 1.8
        QAcondB <- QAAmid
        # Drawing new taus from multivariate normal
        # distribution
        tauProp <- t(chol(solve(QAcondB))) %*% rnorm(N) +
          muAcondB
        # Accept/reject step
        if (runif(1) < acceptProb(rain$n.rain[A], tauI,
          tauProp, rain$n.year[A])) {
          tau[A] <- tauProp
          accepts[i, A] <- TRUE
        }
    }
}
# b = TT
A = tail(stepp, 1):TT
B = -A
QAAb366 <- Q[A, A]
QABb366 <- Q[A, B]
```

```r
    if (length(A) != 1) {
        Qb366 <- solve(QAAb366) %*% QABb366[, tail(stepp,
            1) - 1]
        tauI <- tau[A]
        tauminusI <- tau[B]
        # Defining mean from Eq. 1.7
        muAcondB <- -Qb366[, ncol(Qb366)] * tauminusI[length(tauminusI)]
        # Defining Precision from Eq. 1.8
        QAcondB <- QAAb366
        # Drawing new taus from multivariate normal
        # distribution
        tauProp <- t(chol(solve(QAcondB))) %*% rnorm(length(A)) +
            muAcondB
        # Accept/reject step
        if (runif(1) < acceptProb(rain$n.rain[A], tauI, tauProp,
            rain$n.year[A])) {
            tau[A] <- tauProp
            accepts[i, A] <- TRUE
        }
    } else {
        Qb366 <- solve(QAAb366) %*% QABb366[tail(stepp, 1) -
            1]
        tauI <- tau[A]
        tauminusI <- tau[B]
        # Defining mean from Eq. 1.7
        muAcondB <- -Qb366[, ncol(Qb366)] * tauminusI[length(tauminusI)]
        # Defining Precision from Eq. 1.8
        QAcondB <- QAAb366
        # Drawing new taus from multivariate normal
        # distribution
        tauProp <- rnorm(1, muAcondB, sqrt(solve(QAcondB)))
        # Accept/reject step
        if (runif(1) < acceptProb(rain$n.rain[A], tauI, tauProp,
            rain$n.year[A])) {
            tau[A] <- tauProp
            accepts[i, A] <- TRUE
        }
    }
    tauMat[i + 1, ] <- tau
    sigma2Vec[i] <- sigma2
}
acceptRate <- mean(accepts)
currTime <- proc.time()[3]
print(paste0("Total time taken: ", round((currTime - startTime)/60,
```

```
        digits = 2), " minutes"))
    print(paste0("Acceptance rate: ", round(acceptRate, digits = 4)))
    return(list(taumatrix = tauMat[(burnin + 1):(M + 1), ], sigma2 = sigma2Vec[(burnin +
        1):M], acceptrate = acceptRate))
}
resultsBLOCK <- MCBLOCK(rain, 15)
```

```
## [1] "Total time taken: 1.51 minutes"
## [1] "Acceptance rate: 0.3058"
```

```
tausBLOCK <- resultsBLOCK$taumatrix
sigma2BLOCK <- resultsBLOCK$sigma2
acceptRateBLOCK = resultsBLOCK$acceptrate
cat("Effective sample size of single site update: ", max(effectiveSize(as.mcmc(cbind(tau
    sigma2)))), "\nEffective sample size of block site update : ",
    max(effectiveSize(as.mcmc(cbind(tausBLOCK, sigma2BLOCK)))))
```

```
## Effective sample size of single site update:  3955.381
## Effective sample size of block site update :  11996.39
```

We see in Figure 7 that the autocorrelation goes to zero faster than the single site update do. Also we see in Figure 8 that the differences are equally as small as for the single site update MCMC. Also the running time is much faster for the block update MCMC, so we can conclude that with step size 10, the block algorithm is more efficient than the single site update MCMC.

```
par(mfrow = c(3, 1))
acf(expit(tausBLOCK[, 1]))
acf(expit(tausBLOCK[, 201]))
acf(expit(tausBLOCK[, 366]))
```

```
meansBLOCK <- c()
for (i in 1:366) {
    meansBLOCK[i] <- mean(expit(tausBLOCK[, i]))
}
plot(abs(meansBLOCK - (rain$n.rain/rain$n.years)), type = "l",
    ylim = c(0, 1), xlab = "t")
abline(h = mean(abs(meansBLOCK - (rain$n.rain/rain$n.years))),
    col = "blue")
legend("topright", inset = 0.05, legend = c("error between pi and yt/nt as a function of
    "Mean of the error between pi and yt/nt"), lty = 1, col = c("black",
    "blue"), cex = 0.6)
```

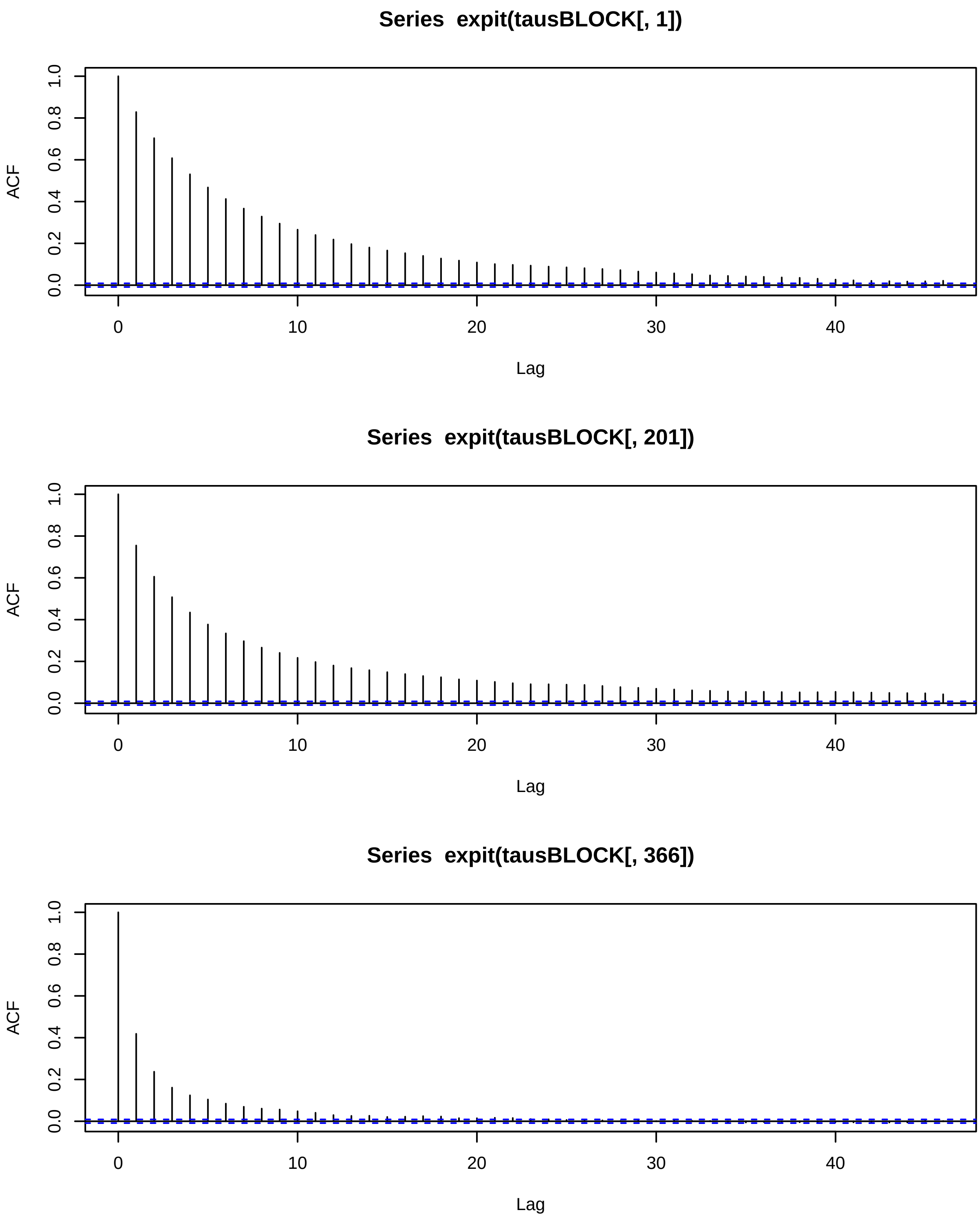


Figure 7: Estimated autocorrelation function for expit of tau_1, tau_{201}, and tau_{366} when using the block site update.

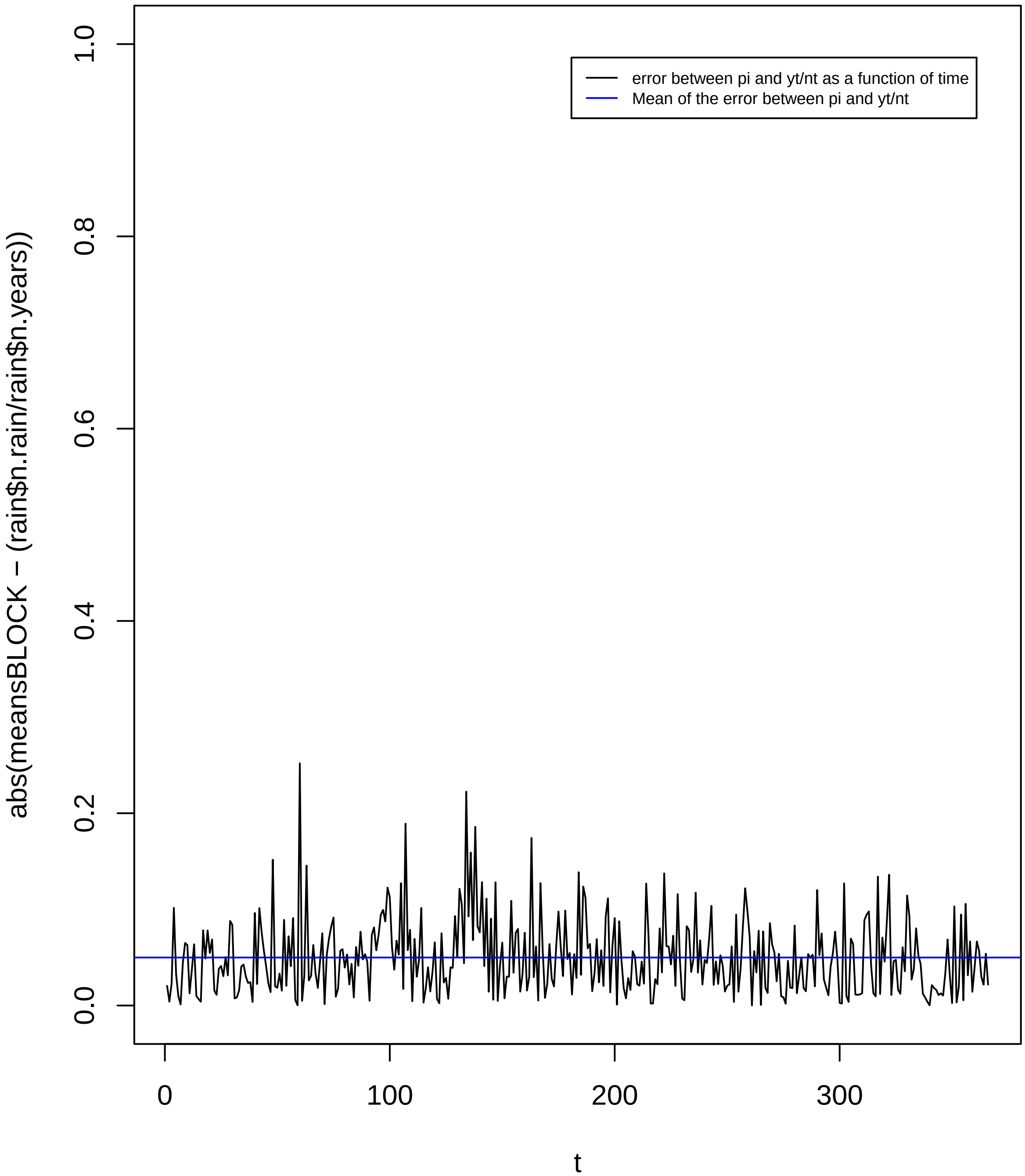


Figure 8: Associated uncertainties of pi and y_t/n_t when using the block site update.

We can also check for other step sizes. We will now test for $N = 2$, $N = 5$, $N = 10$,, $N = 25$, $N = 50$, and $N = 100$. We dont have access to a pc that can test every single step size, so we will just test these 5 to see if we find the lowest running time.

```
resultBlock2 <- MCBLOCK(rain, 2)
```

```
## [1] "Total time taken: 7.29 minutes"
## [1] "Acceptance rate: 0.8436"
```

```
resultBlock5 <- MCBLOCK(rain, 5)
```

```
## [1] "Total time taken: 3.29 minutes"
## [1] "Acceptance rate: 0.6675"
```

```
resultBlock10 <- MCBLOCK(rain, 10)
```

```
## [1] "Total time taken: 1.94 minutes"
## [1] "Acceptance rate: 0.4536"
```

```
resultBlock25 <- MCBLOCK(rain, 25)
```

```
## [1] "Total time taken: 1.31 minutes"
## [1] "Acceptance rate: 0.1351"
```

```
resultBlock25 <- MCBLOCK(rain, 50)
```

```
## [1] "Total time taken: 1.73 minutes"
## [1] "Acceptance rate: 0.0238"
```

```
resultBlock100 <- MCBLOCK(rain, 100)
```

```
## [1] "Total time taken: 3.66 minutes"
## [1] "Acceptance rate: 0.001"
```

We see that a step size between 15 and 25 gives the most efficient algorithm.

Problem B

Introduction

In this problem we will use INLA to analyze the Tokyo rainfall as seen in Problem A.

a)

INLA

```
startTime <- proc.time()[3]
mod <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = F, hyper = list(prec = list(pri
    param = c(2, 0.05)))), family = "binomial", data = rain,
    Ntrials = n.years, control.compute = list(config = T), verbose = F,
    control.inla = list(strategy = "simplified.laplace", int.strategy = "ccd"),
    control.fixed = list(prec.intercept = 1, prec = 1))
currTime <- proc.time()[3]
```

INLA performs random walk of order 1 in the according way, $\Delta x_i \sim \mathcal{N}(0, \nu^{-1})$, where the prior is defined on θ, It is related to ν with the relationship,

$$\theta = log(\nu).$$

Since we have have from problem A that $\nu^{-1} = 1/(\sigma_u^2)$ we can easily reason to what the prior of θ should be to get the same prior as in problem A. Taking the inverse of an inverse gamma distribution becomes gamma distributed. Then the logarithm of a gamma distribution is a log-gamma distribution. We are given the parameters $\alpha = 2$ and $\beta = 0.02$ for the inverse gamma distribution and after the transformation the parameters do not change. Thus the prior should be log-gamma distributed.

We will compare their predictions and uncertainties to what we obtained by MCMC by plotting $\pi(\tau_t)$ VS t and its corresponding 95% confidence interval, as well as the marginal densities of $\pi(\tau_1), \pi(\tau_{201})$, $\pi(\tau_{366})$ and σ_u^2. The $\pi(\tau_t)$ VS t is given in Figure 9. We also see in the same picture that INLA gives the same values as our MCMC samplers.

```
# 1 plotting the mean of marginals pi(tau_t) vs t
plot(NULL, NULL, lwd = 3, ylim = c(0, 0.6), xlim = c(0, 366),
    ylab = "Probability of rain", xlab = "days in a year")
lines(x = c(1:length(mod$summary.fitted.values$mean)), y = c(mod$summary.fitted.values$`
    lty = 2, lwd = 3)
lines(x = c(1:length(mod$summary.fitted.values$mean)), y = c(mod$summary.fitted.values$m
    lwd = 3)
lines(x = c(1:length(mod$summary.fitted.values$mean)), y = c(mod$summary.fitted.values$`
```

```
    lty = 2, lwd = 3)
lines(means, type = "l", lwd = 3, col = "red")
lines(meansBLOCK, type = "l", lwd = 3, col = "blue")
legend("topleft", inset = 0.05, c("INLA with 95% CI", "Single site MCMC",
    "Block site MCMC"), fill = c("black", "red", "blue"), cex = 0.6)
```

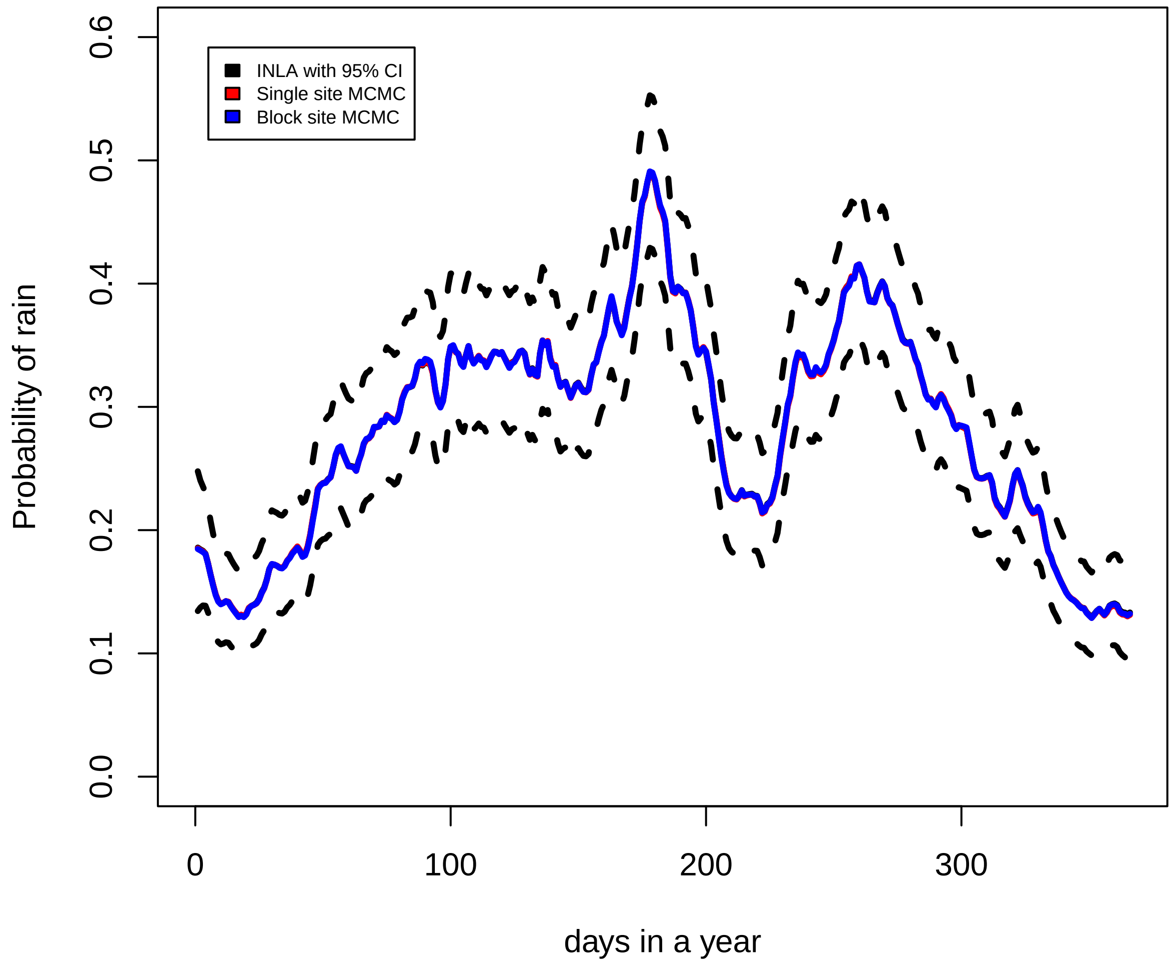


Figure 9: Pi as a function of time using INLA together with its 95 percent C.I. Also we have plottet the samples obtained from our two MCMC algorithms.

The marginal plots of $\pi(\tau_1), \pi(\tau_{201}), \pi(\tau_{366})$ and σ_u^2 are given in Figure 10 respectively. We see that the results are the same as we get in with the MCMC methods.

```
# make this func for simplisity (it is not significant)
mx <- function(list, max) {
    if (max == 1) {
        return(max(list))
    } else {
        return(min(list))
    }
}
par(mfrow = c(2, 2))
### day 1
box_1 <- inla.smarginal(mod$marginals.random$day$index.1, log = FALSE,
    extrapolate = 0, keep.type = FALSE, factor = 15L)
x_1 <- inv.logit(box_1$x)
y_1 <- inv.logit(box_1$y)
# 2
plot(NULL, NULL, ylim = c(mx(y_1, 0), mx(y_1, 1)), xlim = c(mx(x_1,
    0), mx(x_1, 1)), ylab = "Conditional likelihood", xlab = "pi(tau_1)")
lines(x = x_1, y = y_1, col = "black")
abline(v = mod$summary.fitted.values$mean[1], col = "blue", lwd = 3)
abline(v = mod$summary.fitted.values$`0.025quant`[1], col = "blue",
    lwd = 3, lty = 2)
abline(v = mod$summary.fitted.values$`0.975quant`[1], col = "blue",
    lwd = 3, lty = 2)
## day 201
box_201 <- inla.smarginal(mod$marginals.random$day$index.201,
    log = FALSE, extrapolate = 0, keep.type = FALSE, factor = 15L)
x_201 <- inv.logit(box_201$x)
y_201 <- inv.logit(box_201$y)
# 3
plot(NULL, NULL, ylim = c(mx(y_201, 0), mx(y_201, 1)), xlim = c(mx(x_201,
    0), mx(x_201, 1)), ylab = "Conditional likelihood", xlab = "pi(tau_201)")
lines(x = x_201, y = y_201, col = "black")
abline(v = mod$summary.fitted.values$mean[201], col = "blue",
    lwd = 3)
abline(v = mod$summary.fitted.values$`0.025quant`[201], col = "blue",
    lwd = 3, lty = 2)
abline(v = mod$summary.fitted.values$`0.975quant`[201], col = "blue",
    lwd = 3, lty = 2)
## day 356
box_201 <- inla.smarginal(mod$marginals.random$day$index.366,
    log = FALSE, extrapolate = 0, keep.type = FALSE, factor = 15L)
x_356 <- inv.logit(box_201$x)
```

```r
y_356 <- inv.logit(box_201$y)
# 4
plot(NULL, NULL, ylim = c(mx(y_356, 0), mx(y_356, 1)), xlim = c(mx(x_356,
    0), mx(x_356, 1)), ylab = "Conditional likelihood", xlab = "pi(tau_366)")
lines(x = x_356, y = y_356, col = "black")
abline(v = mod$summary.fitted.values$mean[366], col = "blue",
    lwd = 3)
abline(v = mod$summary.fitted.values$`0.025quant`[366], col = "blue",
    lwd = 3, lty = 2)
abline(v = mod$summary.fitted.values$`0.975quant`[366], col = "blue",
    lwd = 3, lty = 2)
### 1/sigma_u^2
box_hyper <- mod$marginals.hyperpar$`Precision for day`
q_1_sig <- mod$summary.hyperpar$`0.025quant`
q_2_sig <- mod$summary.hyperpar$`0.975quant`
q_mean_sig <- mod$summary.hyperpar$mean
x_hyp <- box_hyper[, 1]
y_hyp <- box_hyper[, 2]
# 5
plot(NULL, NULL, ylim = c(mx(y_hyp, 0), mx(y_hyp, 1)), xlim = c(mx(x_hyp,
    0), mx(x_hyp, 1)), ylab = "Conditional likelihood", xlab = "1/sigma_u^2")
lines(x = x_hyp, y = y_hyp, col = "black")
abline(v = q_1_sig, col = "blue", lwd = 3, lty = 2)
abline(v = q_2_sig, col = "blue", lwd = 3, lty = 2)
abline(v = q_mean_sig, col = "blue", lwd = 3)
```

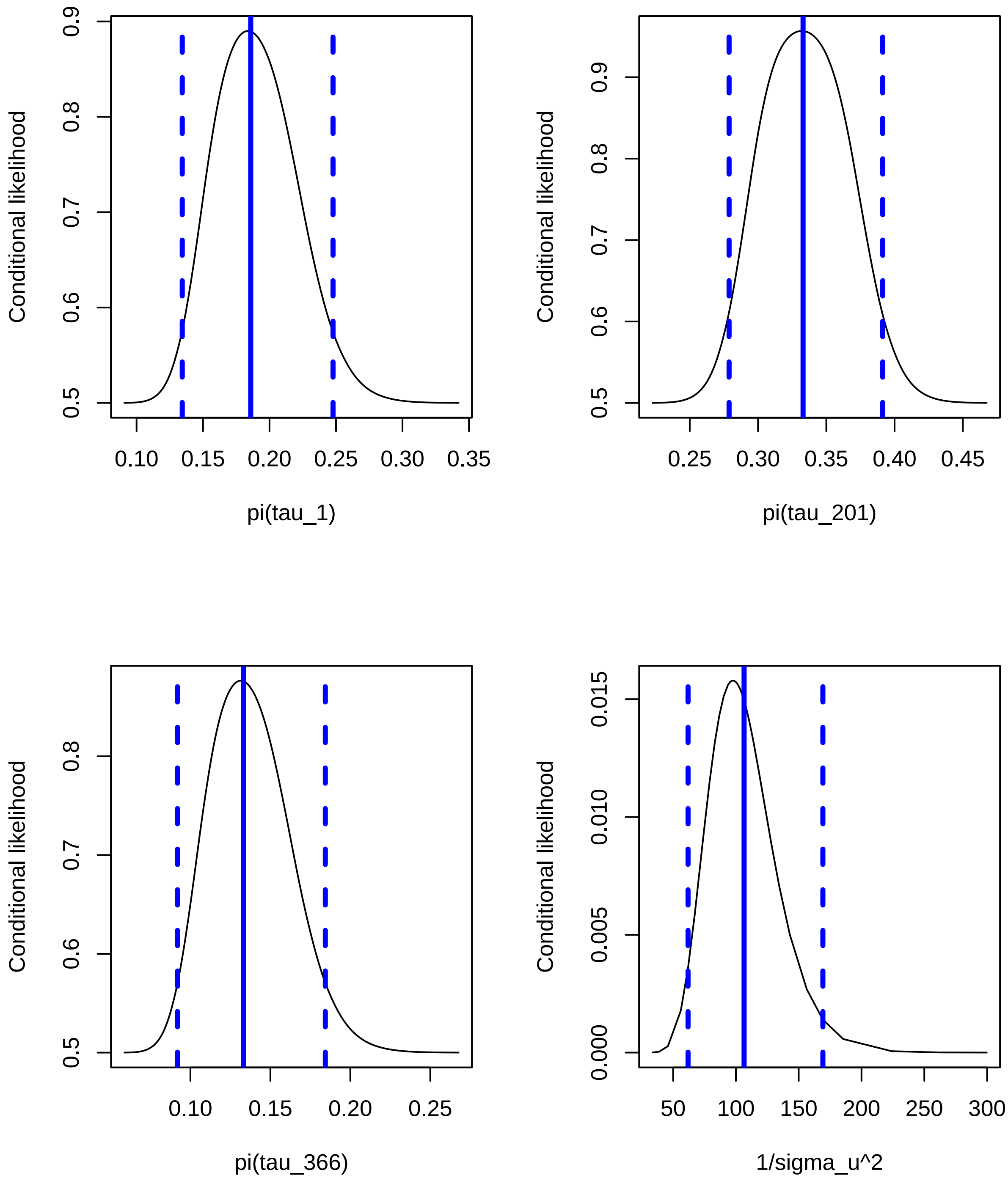


Figure 10: Marginal plots

We can observe that this does indeed coincide with what we have observed when coding the MCMC chain. The only difference here is that the mean is only given for $1/\sigma_u^2$. Table 1 below shows the difference of when MCMC and INLA is applied to make it clearer.

```r
ro <- function(x) {
    return(round(x, 3))
}
# 5
df <- data.frame(Cat = c("1/sigmau2", "Tau1", "Tau201", "Tau366"),
    INLA_Mean = linebreak(c(ro(mod$summary.hyperpar$mean), ro(mod$summary.fitted.values$
        ro(mod$summary.fitted.values$mean[201]), ro(mod$summary.fitted.values$mean[366])
    MCMC_Individual = linebreak(c(round(mean(1/sigma2), 3), mean(expit(taus[,
        1])), mean(expit(taus[, 201])), mean(expit(taus[, 366])))),
    MCMC_Block = linebreak(c(round(mean(1/sigma2BLOCK), 3), mean(expit(tausBLOCK[,
        1])), mean(expit(tausBLOCK[, 201])), mean(expit(tausBLOCK[,
        366])))))
kable(df, col.names = c("", "INLA Mean", "MCMC Individual Mean",
    "MCMC Block Mean"), escape = F, caption = "Mean of marrginaldistributions") %>%
    kable_styling(latex_options = "hold_position")
```

Table 1: Mean of marrginaldistributions

	INLA Mean	MCMC Individual Mean	MCMC Block Mean
1/sigmau2	106.505	105.7090000	103.9870000
Tau1	0.186	0.1852157	0.1846862
Tau201	0.333	0.3337721	0.3342796
Tau366	0.133	0.1312289	0.1320875

Table 2 elow shows the mean of σ_u^2 for the different methods applied. This is done because it is easy to find from $1/\sigma_u^2$.

```r
df <- data.frame(Cat = c("sigmau2"), INLA_Mean = linebreak(c(ro(1/mod$summary.hyperpar$m
    MCMC_Individual = linebreak(c(round(mean(sigma2), 3))), MCMC_Block = linebreak(c(rou
        3))))
kable(df, col.names = c("", "INLA Mean", "MCMC Individual Mean",
    "MCMC Block Mean"), escape = F, caption = "Mean of marrginaldistributions") %>%
    kable_styling(latex_options = "hold_position")
```

Table 2: Mean of marrginaldistributions

	INLA Mean	MCMC Individual Mean	MCMC Block Mean
sigmau2	0.009	0.01	0.01

Looking at the time it took to fit the data set with INLA we can run the code bellow which gives us the time it took.

```
print(paste0("Total time taken: ", round((currTime - startTime)/60,
    digits = 2), " minutes"))
```

```
## [1] "Total time taken: 0.01 minutes"
```

This is much faster than what the single update and block update for MCMC.

b)

We will check how robust the results of the two “control.inla” inputs we have used by changing the methods of the model with different inputs. If we run “?control.inla” we get that the variable “stratagy” can be set to the following: “auto” (default), “gaussian”, “simplified.laplace”, “laplace” or “adaptive”. Further more the variable “int.strategy” can be set to: ‘auto’ (default), ‘ccd’, ‘grid’, ‘eb’ (empirical bayes), ‘user’ or ‘user.std’. We will only show the fit of the model with strategy set to: “auto”, “gaussian”, “simplified.laplace”, “laplace” and “adaptive” with the integration strategy’s: ‘ccd’, ‘grid’ and ‘eb’.

In Figure 11 we can see the mean of the marginals of $\pi(\tau_t)$ VS t for all the different parameters “strategy” can take and “int.strategy” set to “ccd”. Then in the following plot in Figure 12, we show the mean of $\pi(\tau_t)$ VS t for all the different parameters “strategy” can take and “int.strategy” set to “grid”. Lastly in Figure 13, the same is shown however, “int.strategy” is set to “eb”.

We end up with three plots showing the same graph. Thus we can conclude that the inputs to the parameter “control.inla” are very robust, since we do not see a difference of the same model.

```
cols <- brewer.pal(6, "Set2")
control.inla_1 = list(strategy = "auto", int.strategy = "ccd")
mods_1 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_1)
control.inla_2 = list(strategy = "gaussian", int.strategy = "ccd")
mods_2 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_2)
control.inla_3 = list(strategy = "simplified.laplace", int.strategy = "ccd")
mods_3 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_3)
control.inla_4 = list(strategy = "laplace", int.strategy = "ccd")
mods_4 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_4)
control.inla_5 = list(strategy = "adaptive", int.strategy = "ccd")
mods_5 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_5)
plot(NULL, NULL, ylim = c(0, 0.6), xlim = c(0, 356), ylab = "Probability of rain",
    xlab = "day in a year", main = "int.strategy = ccd")
lines(x = c(1:length(mods_1$summary.fitted.values$mean)), y = c(mods_1$summary.fitted.va
    col = cols[1])
lines(x = c(1:length(mods_2$summary.fitted.values$mean)), y = c(mods_2$summary.fitted.va
    col = cols[2])
lines(x = c(1:length(mods_3$summary.fitted.values$mean)), y = c(mods_3$summary.fitted.va
    col = cols[3])
lines(x = c(1:length(mods_4$summary.fitted.values$mean)), y = c(mods_4$summary.fitted.va
    col = cols[4])
lines(x = c(1:length(mods_5$summary.fitted.values$mean)), y = c(mods_5$summary.fitted.va
    col = cols[5])
```

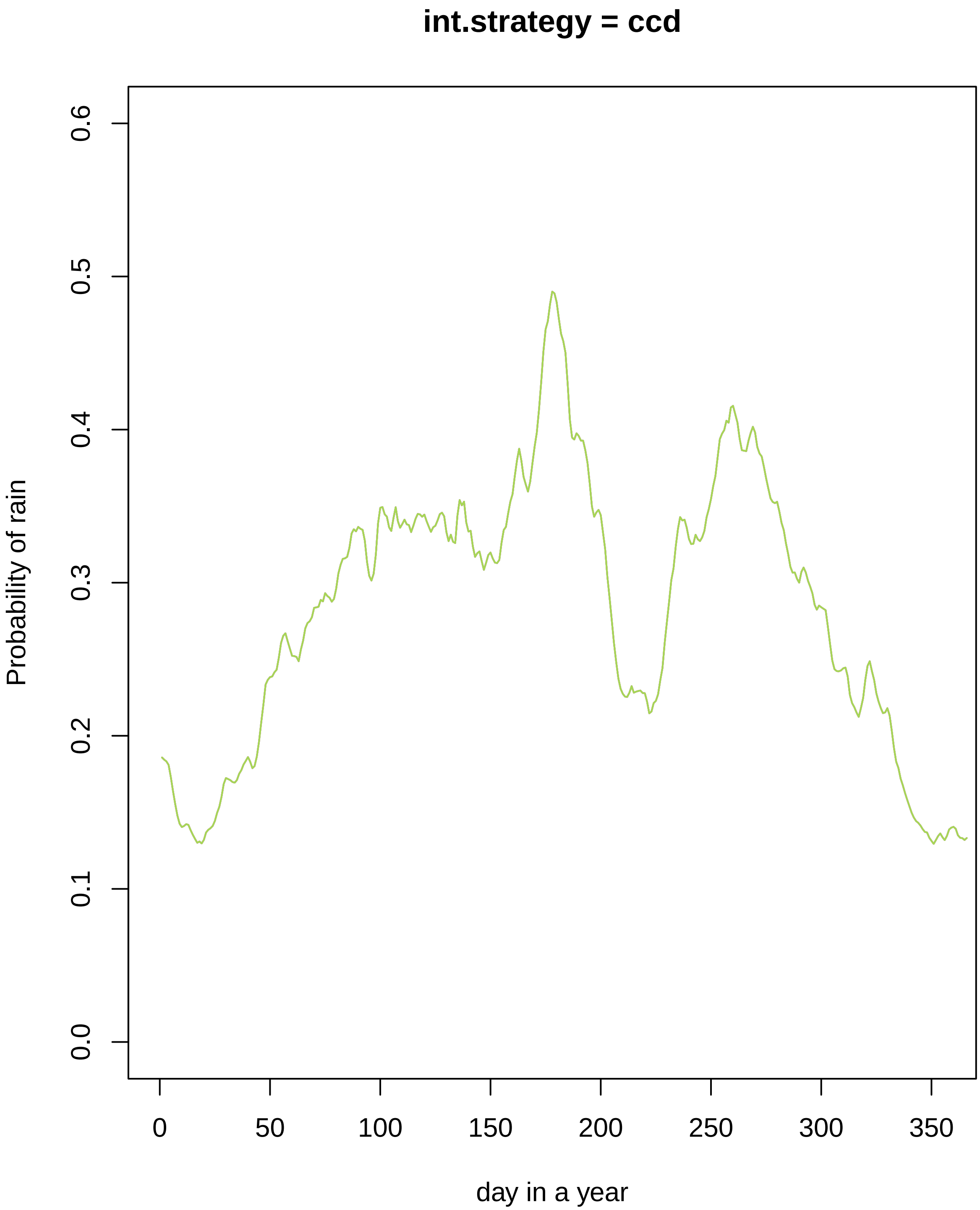


Figure 11: Using strategy CCD

```
cols <- brewer.pal(6, "Set2")
control.inla_1 = list(strategy = "auto", int.strategy = "grid")
mods_1 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_1)
control.inla_2 = list(strategy = "gaussian", int.strategy = "grid")
mods_2 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_2)
control.inla_3 = list(strategy = "simplified.laplace", int.strategy = "grid")
mods_3 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_3)
control.inla_4 = list(strategy = "laplace", int.strategy = "grid")
mods_4 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_4)
control.inla_5 = list(strategy = "adaptive", int.strategy = "grid")
mods_5 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_5)
plot(NULL, NULL, ylim = c(0, 0.6), xlim = c(0, 356), ylab = "Probability of rain",
    xlab = "day in a year", main = "int.strategy = grid")
lines(x = c(1:length(mods_1$summary.fitted.values$mean)), y = c(mods_1$summary.fitted.va
    col = cols[1])
lines(x = c(1:length(mods_2$summary.fitted.values$mean)), y = c(mods_2$summary.fitted.va
    col = cols[2])
lines(x = c(1:length(mods_3$summary.fitted.values$mean)), y = c(mods_3$summary.fitted.va
    col = cols[3])
lines(x = c(1:length(mods_4$summary.fitted.values$mean)), y = c(mods_4$summary.fitted.va
    col = cols[4])
lines(x = c(1:length(mods_5$summary.fitted.values$mean)), y = c(mods_5$summary.fitted.va
    col = cols[5])
```

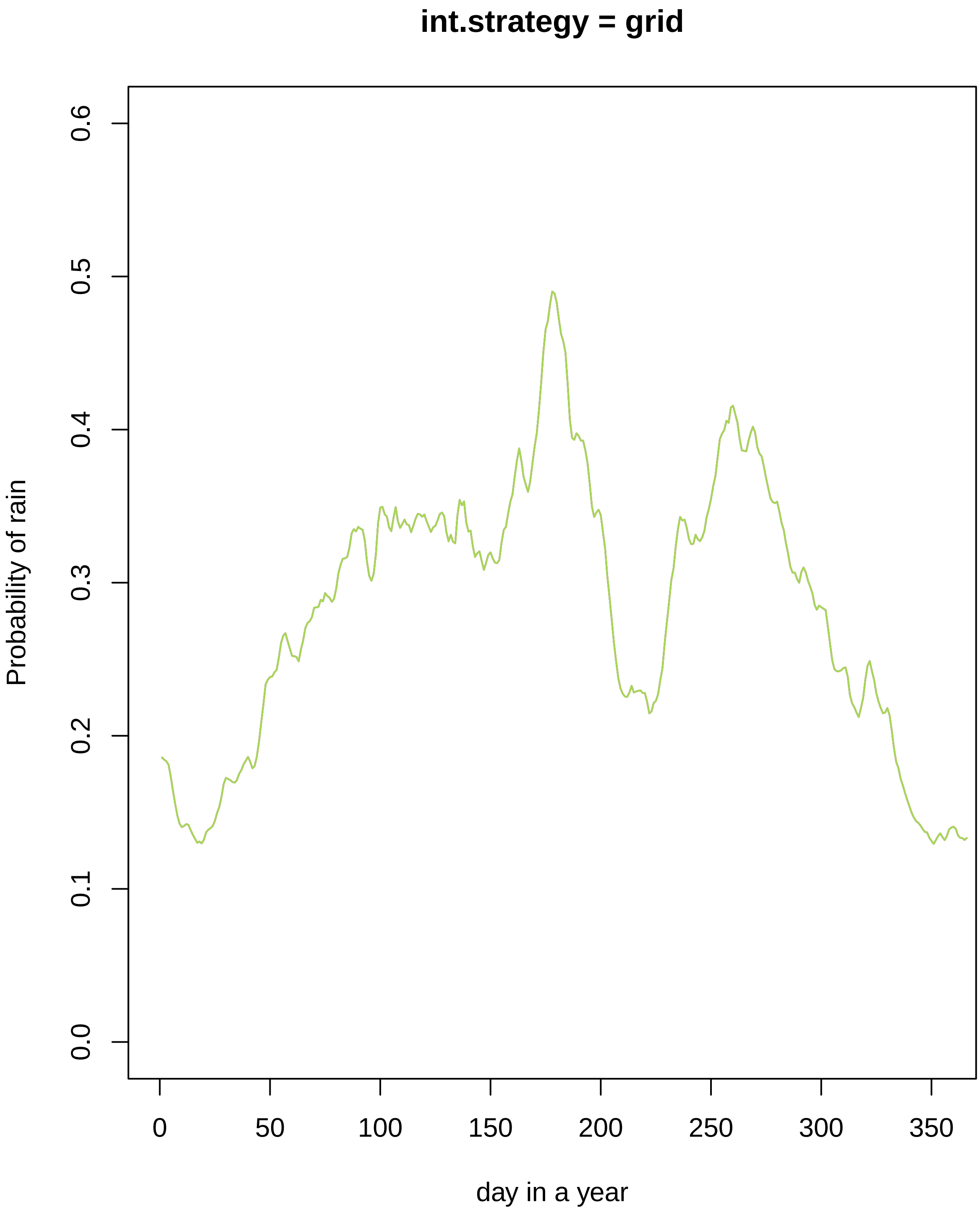


Figure 12: Using strategy Grid

```r
cols <- brewer.pal(6, "Set2")
control.inla_1 = list(strategy = "auto", int.strategy = "eb")
mods_1 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_1)
control.inla_2 = list(strategy = "gaussian", int.strategy = "eb")
mods_2 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_2)
control.inla_3 = list(strategy = "simplified.laplace", int.strategy = "eb")
mods_3 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_3)
control.inla_4 = list(strategy = "laplace", int.strategy = "eb")
mods_4 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_4)
control.inla_5 = list(strategy = "adaptive", int.strategy = "eb")
mods_5 <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2,
        0.05)))), verbose = FALSE, data = rain, Ntrials = n.years,
    control.compute = list(config = TRUE), family = "binomial",
    control.inla = control.inla_5)
plot(NULL, NULL, ylim = c(0, 0.6), xlim = c(0, 356), ylab = "Probability of rain",
    xlab = "day in a year", main = "int.strategy = eb")
lines(x = c(1:length(mods_1$summary.fitted.values$mean)), y = c(mods_1$summary.fitted.va
    col = cols[1])
lines(x = c(1:length(mods_2$summary.fitted.values$mean)), y = c(mods_2$summary.fitted.va
    col = cols[2])
lines(x = c(1:length(mods_3$summary.fitted.values$mean)), y = c(mods_3$summary.fitted.va
    col = cols[3])
lines(x = c(1:length(mods_4$summary.fitted.values$mean)), y = c(mods_4$summary.fitted.va
    col = cols[4])
lines(x = c(1:length(mods_5$summary.fitted.values$mean)), y = c(mods_5$summary.fitted.va
    col = cols[5])
```

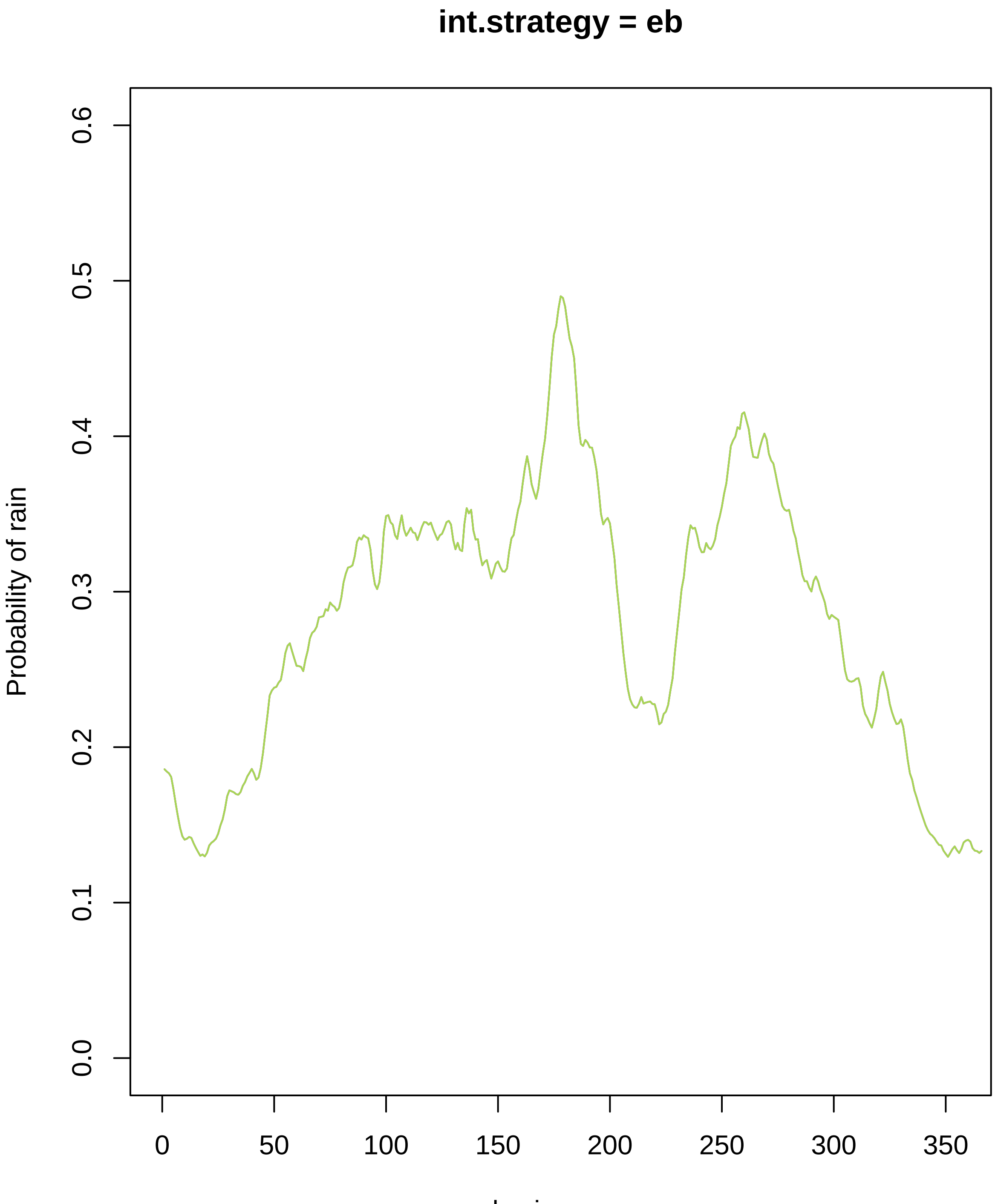


Figure 13: Using strategy eb

c)

We will now apply a different model to the same data set with the same prior. The model that we fit now is only different by an included intercept to the random walk ϵ such that we end up with $\epsilon + \beta$ and that the sum of ϵ has to be zero. This is done by changing "constr=TRUE" and removing "-1" in front of the function. Note that random walk of order 1 is still being preformed.

```
control.inla = list(strategy = "simplified.laplace", int.strategy = "ccd")
mod_new <- inla(n.rain ~ f(day, model = "rw1", constr = TRUE),
    data = rain, Ntrials = n.years, control.compute = list(config = TRUE),
    family = "binomial", verbose = FALSE, control.inla = control.inla)
cat("The intercept is", round(mod_new$summary.fixed$mean, 3))
```

```
## The intercept is -0.986
```

As we can see the intercept is calculated to be -0.986. In Figure 14 we have plotted the model used in problem A (a) in black and the model fit with an intercept in orange.

```
# 10
plot(NULL, NULL, ylim = c(0, 0.6), xlim = c(0, 356), ylab = "Probability of rain",
    xlab = "days in a year")
lines(x = c(1:length(mod_new$summary.fitted.values$mean)), y = c(mod_new$summary.fitted.
    col = "red")
lines(x = c(1:length(mod$summary.fitted.values$mean)), y = c(mod$summary.fitted.values$m
legend("topleft", inset = 0.05, c("with intersept", "without intersept"),
    fill = c("orange", "black"), cex = 0.6)
```

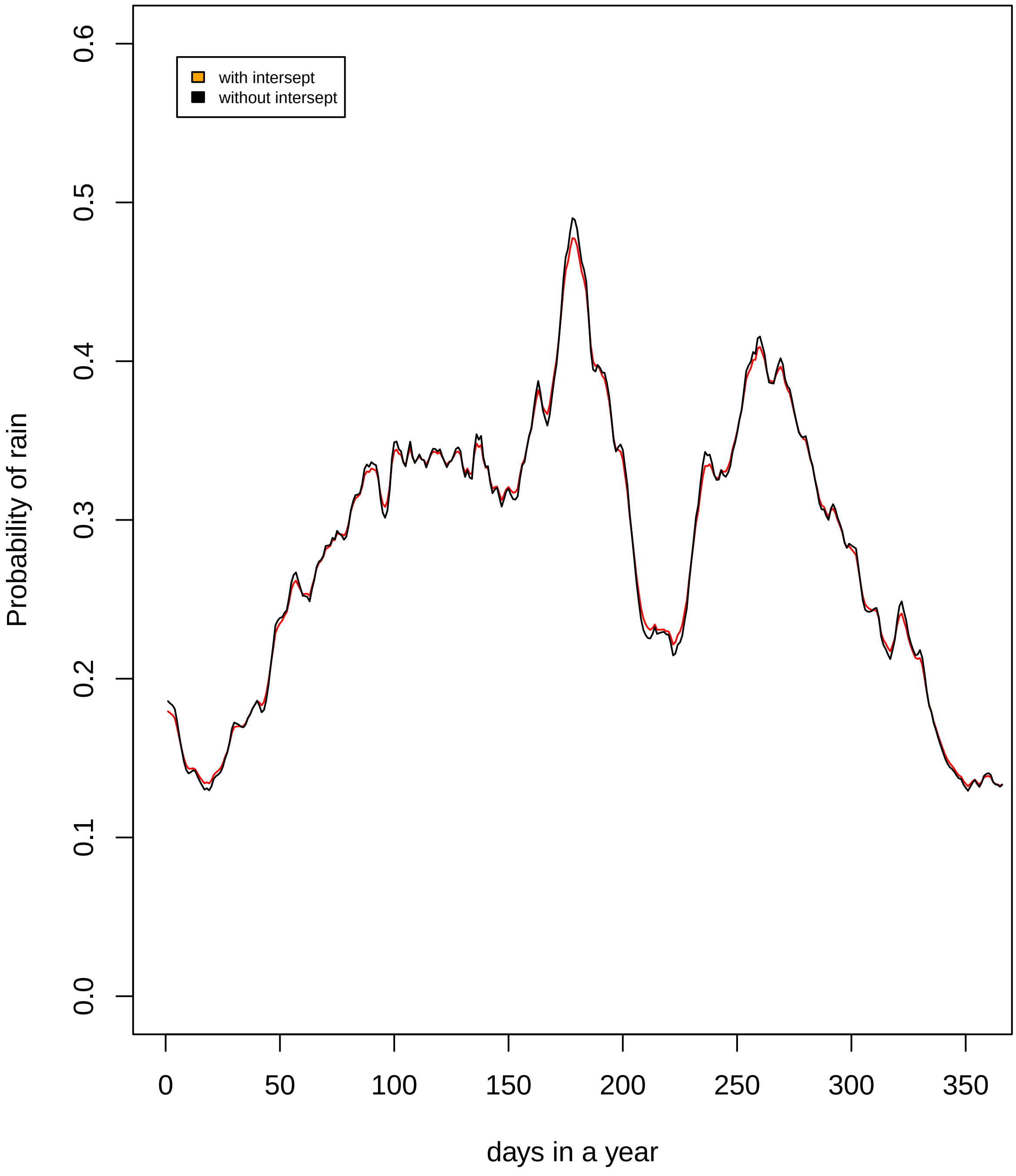


Figure 14: Plotting the new INLA model together with the old one in Problem B: a).

This plot shows that the two models produce posterior which has relatively similar mean. To make this even clearer we will plot the difference of the $\pi(\epsilon_t + \beta)$ and $\pi(\tau_t)$. This is shown in Figure 15.

```
diff <- mod_new$summary.fitted.values$mean - mod$summary.fitted.values$mean
plot(NULL, NULL, ylim = c(mx(diff, 0), mx(diff, 1)), xlim = c(0,
    356), ylab = "Probability of rain", xlab = "days in a year")
lines(x = c(1:length(diff)), y = c(diff), col = "red")
legend("bottomleft", inset = 0.05, c("Difference of the two models"),
    fill = c("red"), cex = 0.6)
```

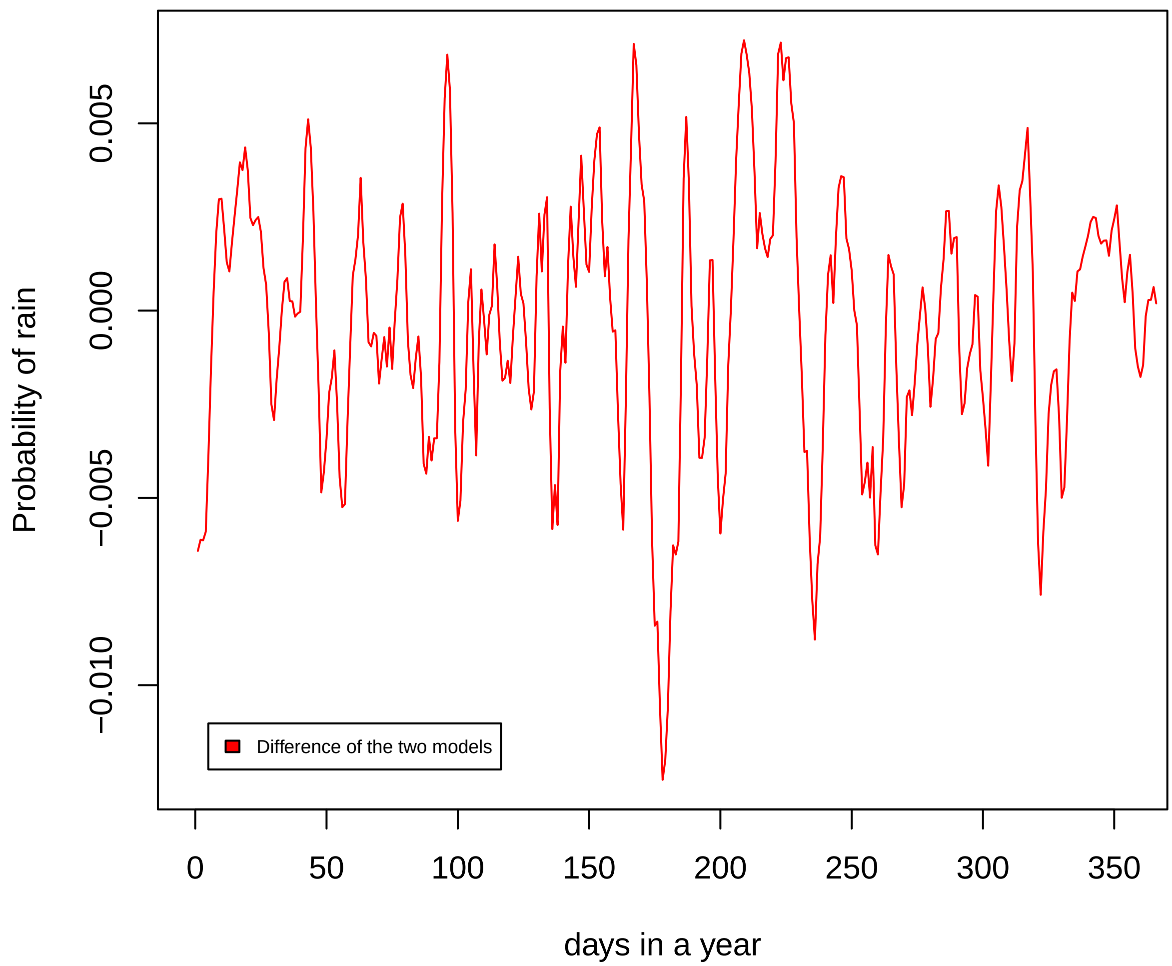
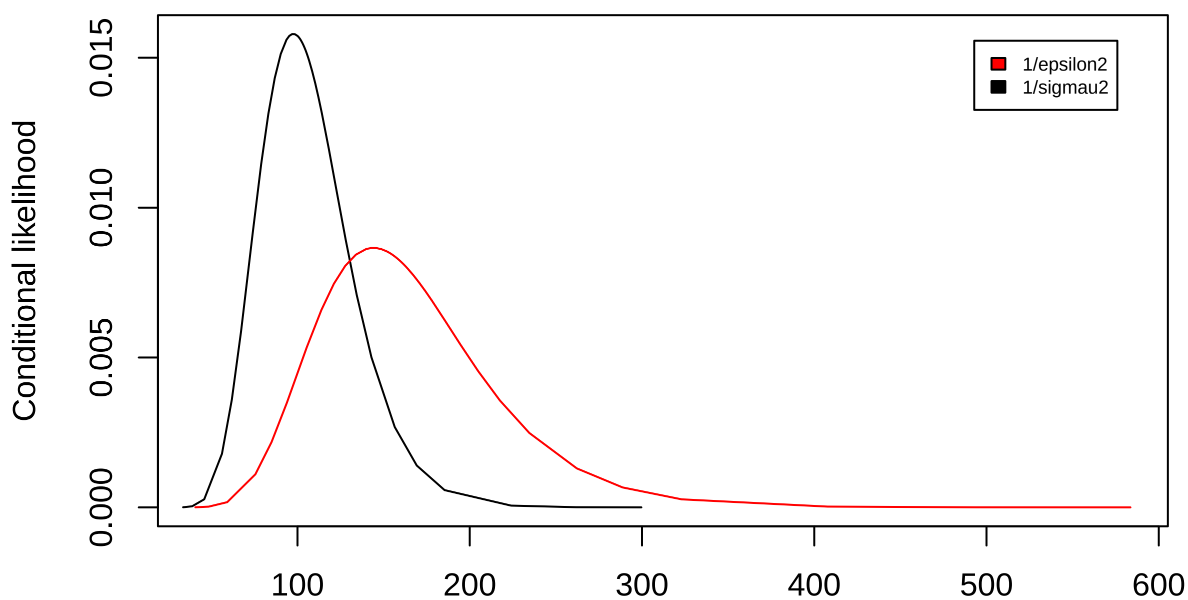


Figure 15: Difference between our two models.

As we can see the differences are minuscule of the predictions of the two models. However when looking at the distribution of $1/\sigma_u^2$ and $1/\epsilon^2$ in plot #######12#########. we can see that they are quite different. Especially we can see that $1/\epsilon^2$ is skewed to the right. This does make sense as we have a negative intercept which gives lower values after the logit function is applied.

```
box_hyper_1 <- mod_new$marginals.hyperpar$`Precision for day`
x_hyp_1 <- box_hyper_1[, 1]
y_hyp_1 <- box_hyper_1[, 2]
plot(NULL, NULL, ylim = c(mx(y_hyp_1, 0), mx(y_hyp, 1)), xlim = c(mx(x_hyp_1,
    0), mx(x_hyp_1, 1)), ylab = "Conditional likelihood", xlab = "")
lines(x = x_hyp, y = y_hyp, col = "black")
lines(x = x_hyp_1, y = y_hyp_1, col = "red")
legend("topright", inset = 0.05, c("1/epsilon2", "1/sigmau2"),
    fill = c("red", "black"), cex = 0.6)
```



If we write up the two models as we have defined then,

$$\begin{aligned}&\text{Model without intercept, linear predictor:} \quad && \tau_t \\ &\text{Model with intercept, linear predictor:} \quad && \epsilon_t + \beta,\end{aligned}$$

where the restriction that the sum of all ϵ_t has to equal zero. We can more clearly see that the two will produce similar results since the posterior of the model with intercept is also dependent upon β and will sett it to the mean of the random walks that were preformed with τ_t. This is reflected upon when calculating the mean of the random walks of the model without intercept:

```
cat("Mean of random walks of model without intercept:", round(mean(mod$summary.random$da
    5), "\nThe intercept in the model with intercept:      ",
    round(mod_new$summary.fixed$mean, 5))
```

```
## Mean of random walks of model without intercept: -0.98601
## The intercept in the model with intercept:       -0.9856
```

We can see that they are almost the same. Thus we end up with a model that is quite similar.