Project2 - TMA4300

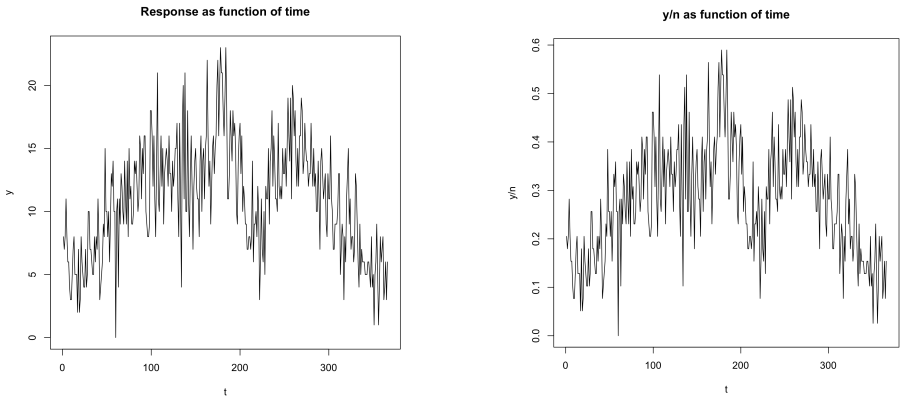
Erik Mjaanes, Sondre Rogde, Eirik Drage Steen

March 2024

1 MCMC

a)

The dataset contains daily observations on rainfall in Tokyo. There are two series: n.years (n) and n.rain (y). n.years counts the number of times a given date has been observed, for example $n_1 = 39$ since January 1st was observed 39 times. $n_t = 39$ for all t except $n_{60} = 10$, corresponding to February 29th. y counts the number of rainy observations (more than 1mm of rain) on a given date. For example $y_1 = 8$ since it rained 8 times on January 1st during the observation period. Figure 1a shows the response against time and 1b shows the proportion of days that were rainy.



(a) Response against time (b) Proportion of rainy days

Figure 1: Data from the Tokyo rainfall dataset

There are two peaks in the proportion of rainy days, around day 180 (June 28th) and 280 (October 6th). The proportion is low around New Year.

The series appear to be jagged, but here is still significant autocorrelation between the proportions of rainy days. Figure 2 shows the ACF for the series y/n.

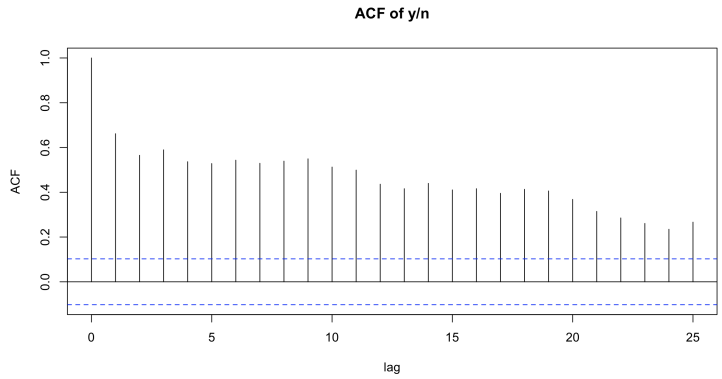


Figure 2: Autocorrelation function for y/n

This is as expected - for example if it often rains around October 6th we expect the probability of rain on October 7th to also be high, since the weather doesn't know exacly what date it is.

b)

Because the $y_t|x_t$ are conditionally independent, we can factor the likelihood function:

$$p(y_1, ..., y_{366}|\pi(x_1), ..., \pi(x_{366})) = \prod_{t=1}^{366} p(y_t|\pi(x_t))$$

Next, since $\pi(\cdot)$ is one-to-one we have that $p(y_t|x_t) = p(y_t|\pi(x_t))$ such that this equals

$$\prod_{t=1}^{366} \binom{n_t}{y_t} (\pi(x_t))^{y_t}(1 - \pi(x_t))^{n_t - y_t}$$

c)

The conditional probability is proportional to the joint probability, such that

$$\begin{aligned} p(\sigma_u^2|\mathbf{y}, \mathbf{x}) &\propto p(\sigma_u^2, \mathbf{y}, \mathbf{x}) \\ &\propto p(\mathbf{y}|\mathbf{x}, \sigma_u^2)p(\mathbf{x}|\sigma_u^2)p(\sigma_u^2) \\ &= p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\sigma_u^2)p(\sigma_u^2) \\ &\propto p(\mathbf{x}|\sigma_u^2)p(\sigma_u^2) \end{aligned}$$

In going from the second to third line we have used that $p(\mathbf{y}|\mathbf{x}, \sigma_u^2) = p(\mathbf{y}|\mathbf{x})$ since the dependence of $\mathbf{y}$ on σ_u^2 is only indirect through $\mathbf{x}$. In going from the third to the fourth line we have used that $p(\mathbf{y}|\mathbf{x})$ does not depend on σ_u^2, so we can absorb this term into the constant of proportionality. Further, we then get

$$p(\sigma_u^2|\mathbf{y}, \mathbf{x}) \propto \left(\prod_{t=2}^{366} \frac{1}{\sigma_u} \exp\left\{ -\frac{1}{2\sigma_u^2}(x_t - x_{t-1})^2 \right\} \right) \left(\frac{\beta^\alpha}{\Gamma(\alpha)} (1/\sigma_u^2)^{\alpha+1} \exp\{-\beta/\sigma_u^2\} \right)$$
$$\propto (1/\sigma_u^2)^{\alpha+1+365/2} \exp\left\{ -\frac{1}{\sigma_u^2} \left(\beta + \frac{1}{2} \sum_{t=2}^{366} (x_t - x_{t-1})^2 \right) \right\}$$

Such that the posterior distribution of σ_u^2 is an inverse gamma distribution with shape parameter $\alpha + 365/2$ and scale parameter $\beta + \frac{1}{2}\sum_{t=2}^{366}(x_t - x_{t-1})^2$.

d)

The acceptance probability with the given proposal distribution is given by

$$\alpha(\mathbf{X}'_{\mathcal{I}}|\mathbf{X}_{-\mathcal{I}}, \sigma_u^2, \mathbf{y}) = \min\left\{1, \frac{p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) Q(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2)}{p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) Q(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2)}\right\}$$
$$= \min\left\{1, \frac{p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2)}{p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2)}\right\}$$

Next, conditioning $\mathbf{y}_{\mathcal{I}}$ on the other parameters in the first term of the numerator and denominator, we get

$$\frac{p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2)}{p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2)} = \frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2)}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2)}$$

And since the $\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}}$ are conditionally independent, the first term simplifies, and this is equal to

$$\frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}}) p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2)}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}}) p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \mathbf{y}_{-\mathcal{I}}, \sigma_u^2)}$$

Similarly, we can condition $\mathbf{y}_{-\mathcal{I}}$ on the remaining parameters in the last term and simplify as the $\mathbf{y}_{-\mathcal{I}}|x_{-\mathcal{I}}$ are conditionally independent. We then get that this is equal to

$$\frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}}) p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}'_{\mathcal{I}}, X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{y}_{-\mathcal{I}}|X_{-\mathcal{I}})}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}}) p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \sigma_u^2) p(\mathbf{y}_{-\mathcal{I}}|X_{-\mathcal{I}})}$$

Cancelling and expanding the joint probability of the parameters $\mathbf{X}_{\mathcal{I}}, X_{-\mathcal{I}}, \sigma_u^2$, we get that this is equal to

$$\frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}})p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}},\sigma_u^2)p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}},\sigma_u^2)p(X_{-\mathcal{I}},\sigma_u^2)}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}})p(\mathbf{X}'_{\mathcal{I}}|X_{-\mathcal{I}},\sigma_u^2)p(\mathbf{X}_{\mathcal{I}}|X_{-\mathcal{I}},\sigma_u^2)p(X_{-\mathcal{I}},\sigma_u^2)} = \frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}})}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}})}$$

Such that the acceptance probability is

$$\min\left\{1, \frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}})}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}})}\right\}$$

As desired.

e)

We have

$$\begin{aligned} p(X|\sigma_u^2) &\propto \exp\left\{ -\frac{1}{2}\left(\begin{bmatrix} X_A & X_B \end{bmatrix} \begin{bmatrix} Q_{AA} & Q_{AB} \\ Q_{BA} & Q_{BB} \end{bmatrix} \begin{bmatrix} X_A \\ X_B \end{bmatrix} \right) \right\} \\ &= \exp\left\{ -\frac{1}{2}\left(X_A^T Q_{AA} X_A + X_A^T Q_{AB} X_B + X_B^T Q_{BA} X_A + X_B^T Q_{BB} X_B \right) \right\} \end{aligned}$$

Adding and subtracting $X_B^T Q_{BA} Q_{AA}^{-1} Q_{AB} X_B$ we get that this equals

$$\exp\left\{ -\frac{1}{2}\left(X_A^T Q_{AA} X_A + X_A^T Q_{AB} X_B + X_B^T Q_{BA} X_A + X_B^T Q_{BA} Q_{AA}^{-1} Q_{AB} X_B X_B^T Q_{BB} X_B - X_B^T Q_{BA} Q_{AA}^{-1} Q_{AB} X_B \right) \right\}$$

Next, since the conditional density is proportional to the joint density:

$$p(X_A|X_B,\sigma_u^2) \propto \exp\left\{ -\frac{1}{2}\left(X_A^T Q_{AA} X_A + X_A^T Q_{AB} X_B + X_B^T Q_{BA} X_A + X_B^T Q_{BA} Q_{AA}^{-1} Q_{AB} X_B \right) \right\}$$

Where the last two terms have been absorbed into the constant of proportionality since they only depend on X_B which we are conditoning on. Next, since Q is a precision matrix and hence symmetric, we have

$$\begin{aligned} Q_{AA}^T &= Q_{AA} \\ (Q_{AA}^{-1})^T &= Q_{AA}^{-1} \\ Q_{AB}^T &= Q_{BA} \end{aligned}$$

And we can write that the conditional density is proportional to

$$\begin{aligned} &\exp\left\{ -\frac{1}{2}\left(X_A^T Q_{AA} X_A + X_A^T Q_{AA} Q_{AA}^{-1} Q_{AB} X_B + X_B^T Q_{BA} Q_{AA}^{-1} Q_{AA} X_A + X_B^T Q_{BA} Q_{AA}^{-1} Q_{AA} Q_{AA}^{-1} Q_{AB} X_B \right) \right\} \\ = &\exp\left\{ -\frac{1}{2}\left(X_A^T Q_{AA} X_A + X_A^T Q_{AA}(Q_{AA}^{-1} Q_{AB} X_B) + (Q_{AA}^{-1} Q_{AB} X_B)^T Q_{AA} X_A + (Q_{AA}^{-1} Q_{AB} X_B)^T Q_{AA}(Q_{AA}^{-1} Q_{AB} X_B) \right) \right\} \\ = &\exp\left\{ -\frac{1}{2}\left((X_A + Q_{AA}^{-1} Q_{AB} X_B)^T Q_{AA}(X_A + Q_{AA}^{-1} Q_{AB} X_B) \right) \right\} \end{aligned}$$

Hence the conditional distribution is a multivariate normal distribution with mean $-Q_{AA}^{-1}Q_{AB}X_B$ and precicion matrix Q_{AA}.

Next, showing that the conditonal distribution has a proper density, we need to show that $\det(Q_{AA}) \neq 0$, since then zero is not an eigenvalue. We first assume A contains X_1 and does not contain X_{366}, and then compute

$$\begin{vmatrix} 1 & -1 & 0 & ... & 0 \\ -1 & 2 & -1 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & ... & -1 & 2 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & ... & 0 \\ -1 & 1 & -1 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & ... & -1 & 2 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & -1 & ... & 0 \\ -1 & 2 & ... & 0 \\ \vdots & \vdots & \ddots & \\ 0 & ... & -1 & 2 \end{vmatrix}$$

Where in the first step we have added the first column to the second and in the second step we have expanded the determinant along the first row, both of which leave the determinant invariant. The matrices on the first row are of size $|A| \times |A|$ and the matrix on the second row is of size $(|A|-1) \times (|A|-1)$. We can repeat this process a total of $|A|-2$ times and then see that

$$\begin{vmatrix} 1 & -1 & 0 & ... & 0 \\ -1 & 2 & -1 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & ... & -1 & 2 \end{vmatrix} = \begin{vmatrix} 1 & -1 \\ -1 & 2 \end{vmatrix} = 1 \neq 0$$

Next, when A contains X_{366} but not X_1, we can use the same method to com-

pute:

$$\begin{vmatrix} 2 & -1 & 0 & ... & 0 & 0 \\ -1 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & ... & 2 & -1 \\ 0 & 0 & 0 & ... & -1 & 1 \end{vmatrix} = \begin{vmatrix} 2 & -1 & 0 & ... & 0 & 0 \\ -1 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & ... & 1 & -1 \\ 0 & 0 & 0 & ... & 0 & 1 \end{vmatrix}$$

$$= (-1)^n \begin{vmatrix} 2 & -1 & 0 & ... & 0 \\ -1 & 2 & -1 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & ... & 1 \end{vmatrix}$$

Where in the first step we have added the last column to the second-to-last and in the second step we have expanded the determinant along the last row. The matrices on the first row are again of size $|A| \times |A|$ and the matrix on the second row is of size $(|A| - 1) \times (|A| - 1)$. We can repeat this process and get that

$$\begin{vmatrix} 2 & -1 & 0 & ... & 0 & 0 \\ -1 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & ... & 2 & -1 \\ 0 & 0 & 0 & ... & -1 & 1 \end{vmatrix} = 0$$

If and only if

$$\begin{vmatrix} 2 & -1 \\ -1 & 1 \end{vmatrix} = 1 = 0$$

Which is clearly false.

Finally we consider the case when A contains neither X_1 nor X_{366}. In this case, we can let

$$\begin{vmatrix} 2 & -1 & 0 & ... & 0 & 0 \\ -1 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & ... & 2 & -1 \\ 0 & 0 & 0 & ... & -1 & 2 \end{vmatrix} = D_n$$

Where the matrix is of size $n \times n$. Expanding the determinant along the first row twice, we get

$$\begin{aligned} D_n &= 2D_{n-1} + \begin{vmatrix} -1 & -1 & 0 & ... & 0 & 0 \\ 0 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & ... & 2 & -1 \\ 0 & 0 & 0 & ... & -1 & 2 \end{vmatrix} \\ &= 2D_{n-1} - D_{n-2} \end{aligned}$$

This is a linear homogeneous recurrence, with characteristic polynomial $(\lambda-1)^2$, which has a repeated root of 1. The solution is therefore

$$D_n = c_1 + c_2 n$$

With base cases $D_1 = 2 = c_1 + c_2$ and $D_2 = 4 - 1 = 3 = c_1 + 2c_2$, we get $c_1 = c_2 = 1$ such that $D_n = 1 + n \neq 0$.

In all cases we have $\det Q_{AA} \neq 0$, so zero is not an eigenvalue and the conditional distribution is proper.

f)

The conditional prior $p(X_t|\mathbf{x_{-t}}, \sigma_u^2)$ can be computed using equation 1 in the problem set. Since the conditional distribution is proportional to the joint, for $t \notin \{1, 366\}$ we get

$$\begin{aligned} p(x_t|\mathbf{x_{-t}}, \sigma_u^2) &\propto p(\mathbf{x}|\sigma_u^2) \\ &\propto \prod_{j=2}^{366} \frac{1}{\sigma_u} \exp\left\{ -\frac{1}{2\sigma_u^2} \left(x_j - x_{j-1}\right)^2 \right\} \\ &\propto \frac{1}{\sigma_u^2} \exp\left\{ -\frac{1}{2\sigma_u^2} \left(x_{t+1}^2 - 2x_{t+1}x_t + 2x_t^2 - 2x_t x_{t-1} + x_{t-1}^2\right) \right\} \\ &\propto \frac{\sqrt{2}}{\sigma_u} \exp\left\{ -\frac{2}{2\sigma_u^2} \left(x_t^2 - x_t(x_{t+1} + x_{t-1})\right) \right\} \\ &\propto \frac{\sqrt{2}}{\sigma_u} \exp\left\{ -\frac{2}{2\sigma_u^2} \left(x_t - \frac{x_{t+1} + x_{t-1}}{2}\right)^2 \right\} \end{aligned}$$

Where in going from the second to the third line we have absorbed all the terms not dependent on x_t into the constant of proportionality. Therefore we get that $x_t|\mathbf{x_{-t}} \sim \mathcal{N}(\frac{x_{t+1}+x_{t-1}}{2}, \frac{\sigma_u^2}{2})$. This is the proposal distribution for the MCMC

model. For $t \in \{1, 366\}$, we get

$$x_1|\mathbf{x_{-1}} \sim \mathcal{N}(x_2, \sigma_u^2)$$
$$x_{366}|\mathbf{x_{-366}} \sim \mathcal{N}(x_{365}, \sigma_u^2)$$

Since only the first and last terms of the product $p(\mathbf{x}|\sigma_u^2)$ respectively remain, leaving terms proportional to the densities of $\mathcal{N}(x_2, \sigma_u^2)$ and $\mathcal{N}(x_{365}, \sigma_u^2)$ respectively. These are therefore the proposal distributions for x_1 and x_{366}. The acceptance probabilities are given by

$$\min\left\{1, \frac{p(y_t|x_t')}{p(y_t|x_t)}\right\}$$

Where $p(y_t|x_t)$ is the binomial density

$$p(y_t|x_t) = \binom{n_t}{y_t} \pi(x_t)^{y_t}(1 - \pi(x_t))^{n_t - y_t}$$

After running the MCMC $50,000$ iterations we get the following results.
Computation time: The total runtime for $50,000$ iterations on a machine using intel core i5 chip is about $429\text{s} \simeq 7.15\text{min}$.
Acceptance rates: The acceptance probability for σ_u^2 is 1 as we are using a Gibbs step for this. Figure 3 displays the acceptance probabilities for each x_t. The mean acceptance rate accross all days is 93.8%.

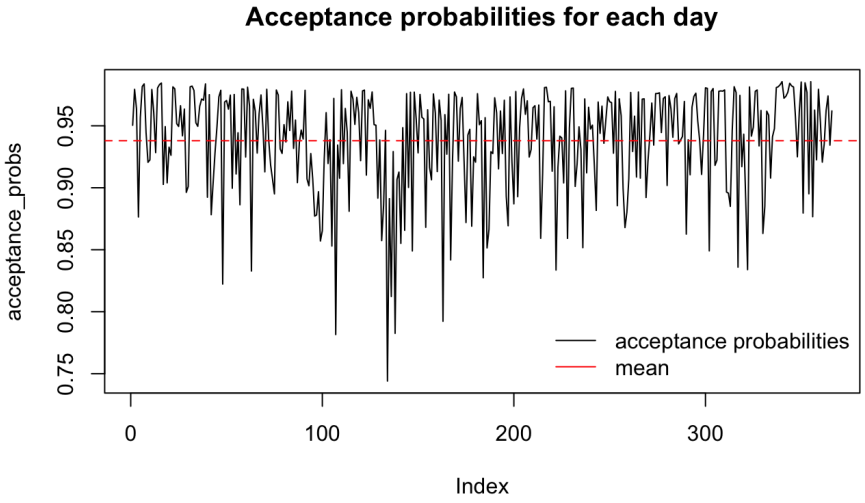


Figure 3: Acceptance rates for each day

Traceplots: figure 4 shows traceplots of x_1, x_{201}, x_{366}, and σ_u^2 including the burnin. The traceplots show that the Markov Chain has converged as the x-values have stabilised around a mean value. The few big values at the beginning

of the traceplots of x_1, x_{201}, and x_{366} is due to the burnin being included in the plots.

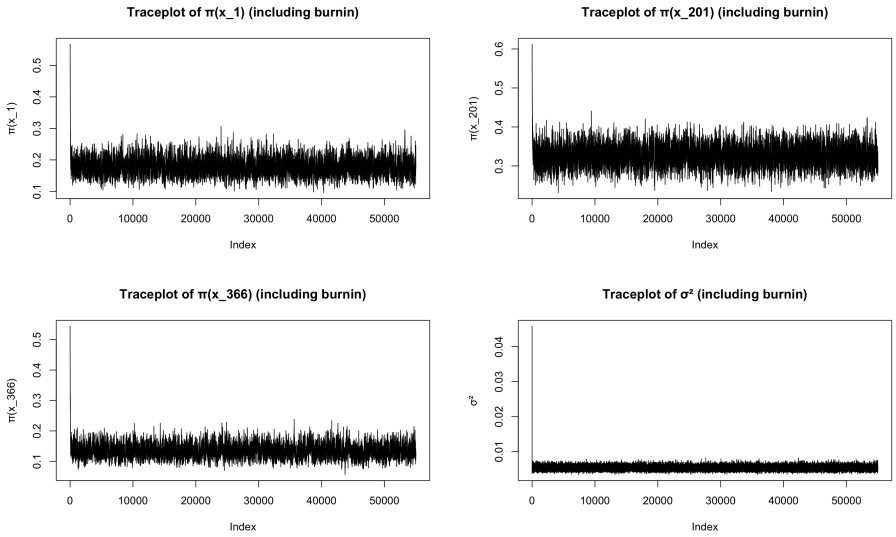


Figure 4: Traceplots for $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

Histograms: figure 5 displays histograms of $\pi(x_1), \pi(x_{201}), \pi(x_{366})$ and σ_u^2 excluding the burnin (first 10% of samples).

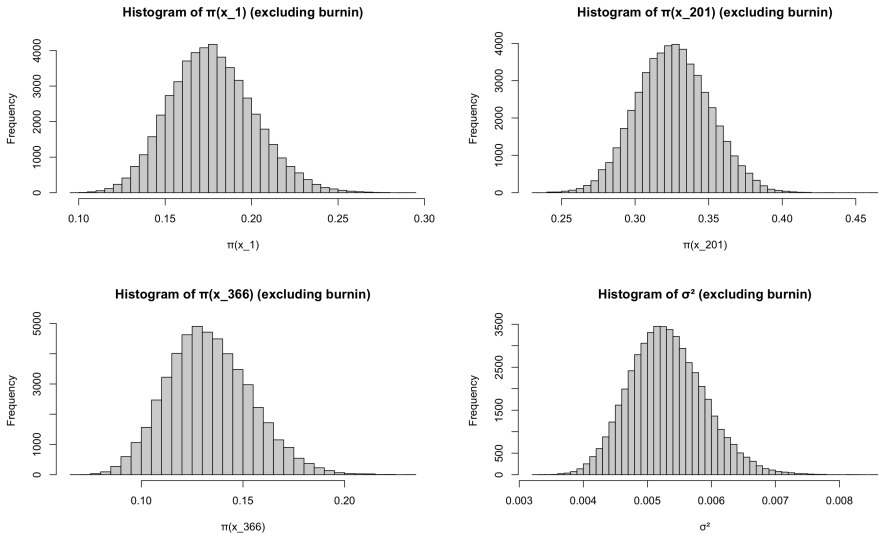


Figure 5: Histograms of $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

ACF: figure 6 displays estimated ACF of $\pi(x_1), \pi(x_{201}), \pi(x_{366})$ and σ_u^2. There is high autocorrelation in the x-values which makes sense as proposals for x_t are correlated with x_{t-1} and x_{t+1}, which we expect to be correlated with the value of x_t in the previous iteration. There is low autocorrelation in the σ^2.

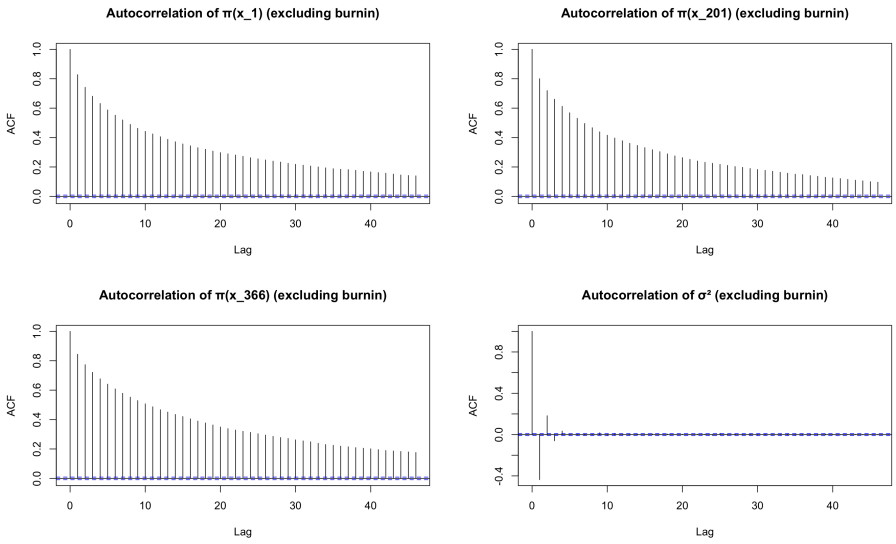


Figure 6: ACF of $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

$\boldsymbol{\sigma_u^2}$: The estimated posterior mean of σ_u^2 is 0.0053 with 95% credible interval $[0.0042, 0.0065]$. The estimated mean precision $(1/\sigma_u^2)$ is 190.7 with 95% credible interval of $[152.8, 235.5]$.

Posterior mean: figure 7 displays the central estimates of $\pi(x_t)$ with 95% credible intervals. The red line plots y_t/n_t, and we can see that the central estimates and credible intervals of $\pi(x_t)$ covers some of this range. Note that the blue interval is not a 95% credible interval for the observations in red, as those are sampled from another distribution (the binomial distribution with parameters n_t and $\pi(x_t)$, scaled by $1/n_t$). We would therefore not expect the red observations to be contained in the credible intervals 95% of the time, and they do not appear to be either.

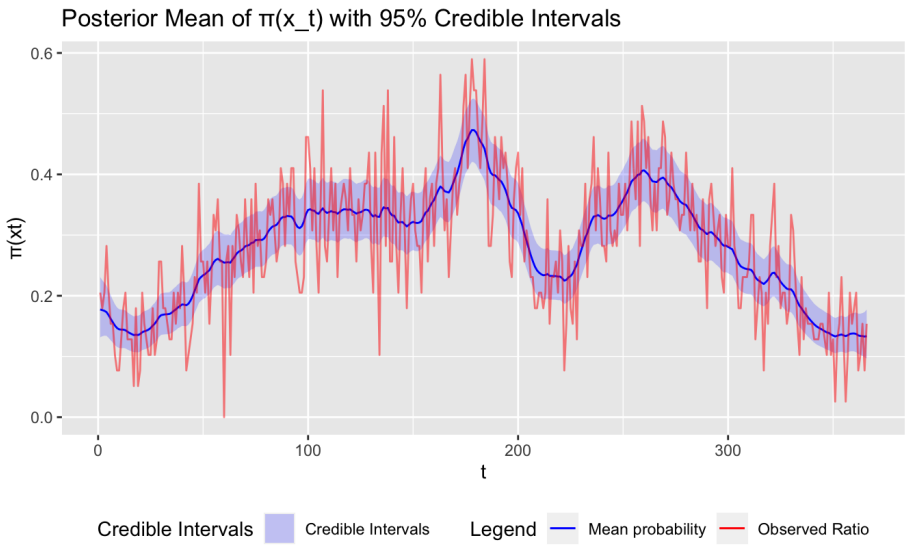


Figure 7: Posterior mean and credible intervals

Below is the code used to arrive at the above results.

```
# declaring parameters
alpha <- 2
beta <- 0.05
iters <- 50000

# loading data
#load(file = "project2/data/rain.rda")
y <- rain[, "n.rain"]
n <- rain[, "n.years"]

# expit function
expit <- function(x) {1/(1+exp(-x))}


mcmc_sampler <- function(alpha, beta, iters, burnin=ceiling(iters/
    10)){
  # declaring variables
  iters = iters + burnin

  # declaring initial data structures
  chain <- matrix(nrow=iters, ncol=367)
  acceptance_probs = matrix(nrow=iters-1, ncol=366)


  # calculating initial x-value
  x <- (y+1)/(n+1)
  pi <- expit(x)

  # sampling first sigma
  sigma_u_sq <- 1/rgamma(1, alpha, beta)
```

```
30
31  # setting initial row for matrix
32  chain[1,] <- c(x, sigma_u_sq)
33
34  # outer loop used for sampling x-vector
35  for (iter in 2:iters){
36    # uniform sampling used for accepting proposals
37    u <- runif(366)
38
39    # vectors for new samples and previous samples respectively
40    new_x = c()
41    old_x = chain[iter-1, ]
42
43    # Gibbs sampling for sigma
44    sigma_alpha <- alpha + 365/2
45    sigma_beta <- beta + 1/2*sum((old_x[2:366] - old_x[1:365])^2)
46    sigma_u_sq <- 1/rgamma(1, shape=sigma_alpha, scale=sigma_beta)
47
48    # inner loop used for sampling x days
49    for (t in 1:366) {
50      # if-statement handling the three cases where t=1, 1<t<366,
    and t=366
51      if (t == 1){
52        proposal <- rnorm(1, old_x[2], sd=sqrt(sigma_u_sq))
53      } else if (t != 366) {
54        proposal <- rnorm(1, (tail(new_x, n=1) + old_x[t+1])/2, sd=
    sqrt(sigma_u_sq/2))
55      } else if (t == 366) {
56        proposal <- rnorm(1, tail(new_x, n=1), sd=sqrt(sigma_u_sq))
57      }
58
59      # calculating new and old pi
60      new_pi = expit(proposal)
61      pi <- expit(old_x[t])
62
63      # calculating acceptance probability (alpha)
64      acceptance_prob <- min(1, dbinom(y[t], n[t], new_pi)/dbinom(y
    [t], n[t], pi))
65
66      # accepting proposal based on acceptance probability
67      new_x <- c(new_x, ifelse(u[t] < acceptance_prob, proposal,
    old_x[t]))
68
69      # used to track observed acceptance probabilities
70      acceptance_probs[iter-1, t] <- (u[t] < acceptance_prob)
71    }
72
73    # updates chain and time data structures
74    chain[iter, ] <- c(new_x, sigma_u_sq)
75
76    if (iter %% 100 == 0){print(iter/iters)}
77  }
78
79  # calculates mean acceptance probabilities
80  acceptance_probs <- colMeans(acceptance_probs)
81
82  # declares list to return
```

```
  result_list <- list(
    chain = chain,
    acceptance_probs = acceptance_probs
  )
  return (result_list)
}

# calls the function and measures the runtime
start_time <- proc.time()[3]
results <- mcmc_sampler(alpha, beta, iters)
end_time <- proc.time()[3]
total_time = end_time - start_time
print(total_time)
```

Listing 1: Code for 1f

g)

The proposed parameter values are here drawn from the conditional multivariate normal derived in problem e). When calculating the acceptance probability for a proposed vector update, we can compute:

$$\begin{aligned}
\frac{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}'_{\mathcal{I}})}{p(\mathbf{y}_{\mathcal{I}}|\mathbf{X}_{\mathcal{I}})} &= \frac{\prod_{j\in\mathcal{I}} \binom{n_j}{y_j} (\pi(x'_j))^{y_j}(1-\pi(x'_j))^{n_j-y_j}}{\prod_{k\in\mathcal{I}} \binom{n_k}{y_k} (\pi(x_k))^{y_k}(1-\pi(x_k))^{n_k-y_k}} \\
&= \prod_{j\in\mathcal{I}} \frac{(\pi(x'_j))^{y_j}(1-\pi(x'_j))^{n_j-y_j}}{(\pi(x_j))^{y_j}(1-\pi(x_j))^{n_j-y_j}} \\
&= \exp\left\{ \sum_{j\in\mathcal{I}} y_j(\ln\pi(x'_j) - \ln\pi(x_j)) + (n_j-y_j)(\ln(1-\pi(x'_j)) - \ln(1-\pi(x_j))) \right\}
\end{aligned}$$

This form is used to avoid numerical overflow issues when calculating the acceptance probability.

Choosing M: After exploring a bit with the parameter M, we found a good choice seems to be between 20 and 40. Having larger block sizes significantly speeds up computing time but reduces acceptance probabilities. If acceptance probaiblities are low we need more iterations to get accurate results. Therefore, choosing a too large M might make it more inefficient. Another procedure that slows the code with larger M is the inverting of the precision matrices Q_{AA}. The plots and discussion below are based on $M = 30$.

Computation time: The total runtime for $50,000$ iterations with $M = 30$ on a machine using intel core i5 chip is about 57 seconds. This is significantly lower than the 7 minutes taken when sampling individual x_t, as in **1f**.

Acceptance rates: figure 8 displays the acceptance probabilities for each batch. The mean acceptance rate accross all days is 0.21. This is however

influenced by the big spike in probability for the last batch. The last batch have higher acceptance probability because it consists of fewer days than the others as the length of the last batch is 366 $\mod M = 6$ in this case. Mean acceptance probabilities excluding the last batch is 0.16. The acceptance rate for σ_u^2 is again 1 as we use a Gibbs step.

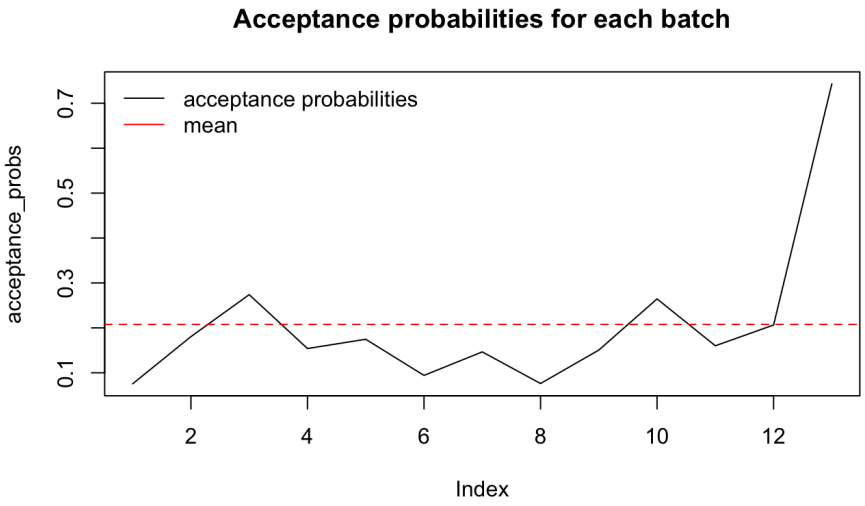


Figure 8: Acceptance rates for each batch

Traceplots: figure 9 shows traceplots of $\pi(x_1), \pi(x_{201}), \pi(x_{366})$ and σ_u^2 including the burnin. The traceplots show that the Markov Chain has converged as the x-values have stabilised around a mean value. The few big values at the beginning of the traceplots of x_1, x_{201}, and x_{366} is due to the burnin being included. Notice that x_1 and x_{201} in 9 have visibly denser oscillations than x_1 and x_{201} in Figure 4 since the batches of 30 have lower acceptance probability than batches of 1, and hence longer stretches of constant sample values. x_{366} does not exhibit this visible difference as the last batch is of size 6.

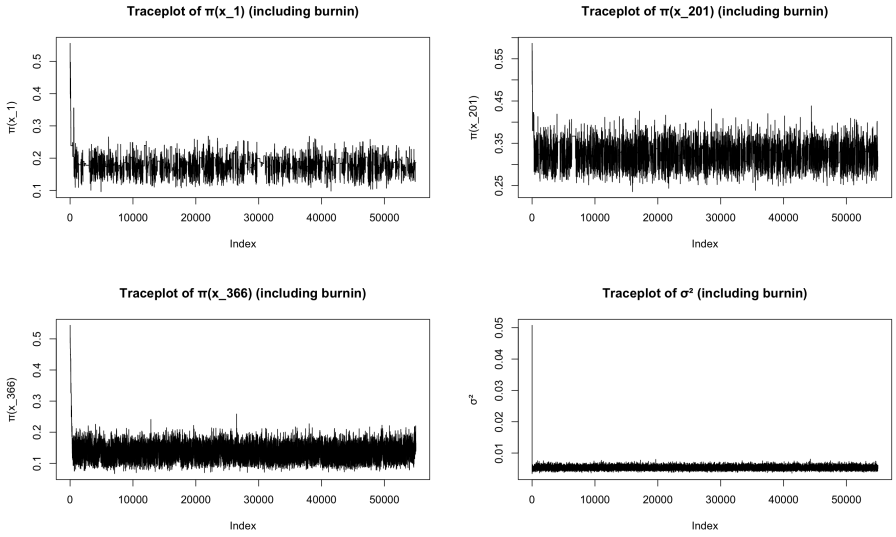


Figure 9: Traceplots for $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

Histograms: figure 10 displays histograms of $\pi(x_1), \pi(x_{201}), \pi(x_{366})$ and σ_u^2 excluding the burnin. Notice that the histograms in figure 10 are close in shape to the histograms in figure 5, but are less smooth. This is likely an effect of the lower acceptance probability leading to more samples in certain buckets if multiple proposals in a row are rejected. This also explains why the histogram for $\pi(x_{366})$ is similar in figures 5 and 10, since x_{366} is in the last batch and has a high acceptance probability, whereas th histograms for $\pi(x_1)$ look more distinct.

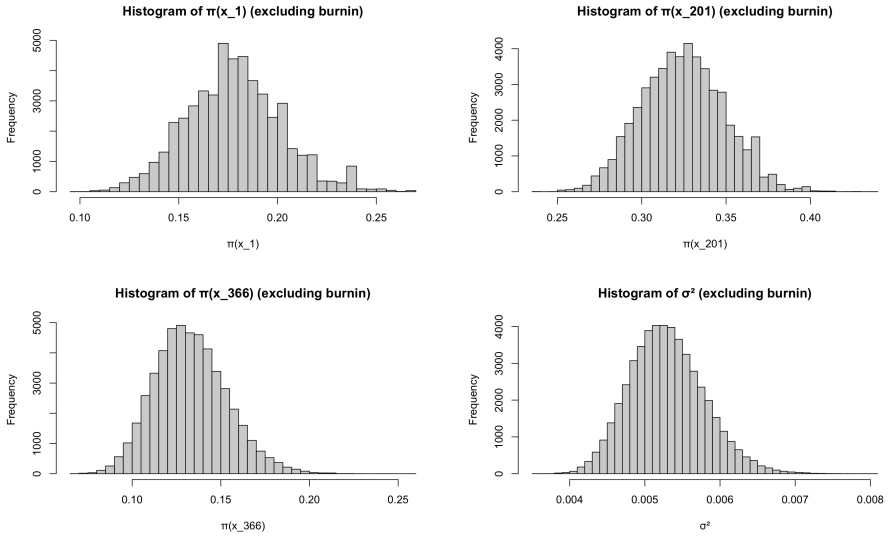


Figure 10: Histograms of $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

ACF: figure 11 displays the ACF of $\pi(x_1), \pi(x_{201}), \pi(x_{366})$ and σ_u^2. There is high autocorrelation in the x-values and low autocorrelation in the σ^2. This again is as expected as proposals for x_t are correlated with the values of x_{t-1} and x_{t+1}, which are correlated with the previous value of x_t.

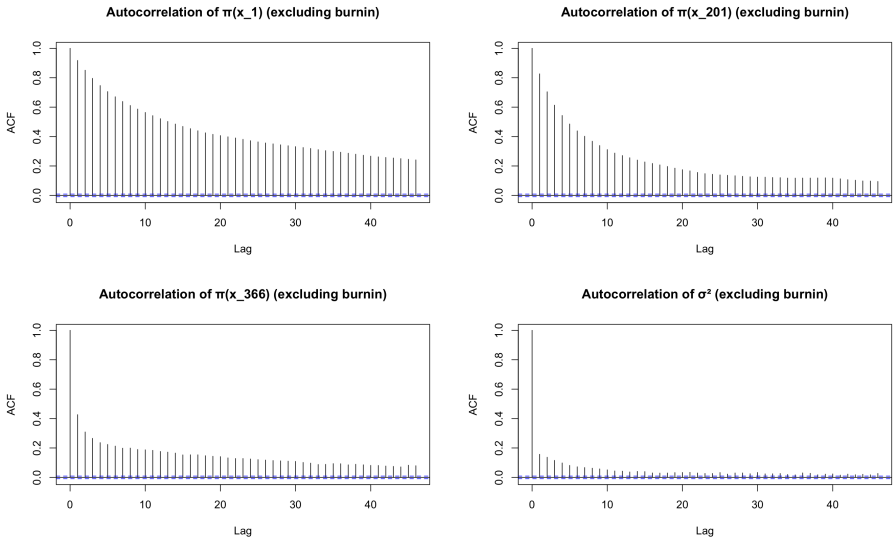


Figure 11: ACF of $\pi(x_1), \pi(x_{201}), \pi(x_{366}), \sigma_u^2$

$\sigma_{\mathbf{u}}^{\mathbf{2}}$: The empirical mean of σ_u^2 is 0.0053 with credible interval $[0.0044, 0.0063]$. The empirical mean precision is 190.4 with 95% credible interval $[157.5, 227.1]$. This is close to the results in **1f**.

Posterior mean: figure 12 displays the central estimates of $\pi(x_t)$ with 95% credible intervals. The red line plots y_t/n_t, and we can see that the central estimates and credible intervals of $\pi(x_t)$ covers some of this range. Note that the blue interval credible interval is close to the credible intreval in **1f**.

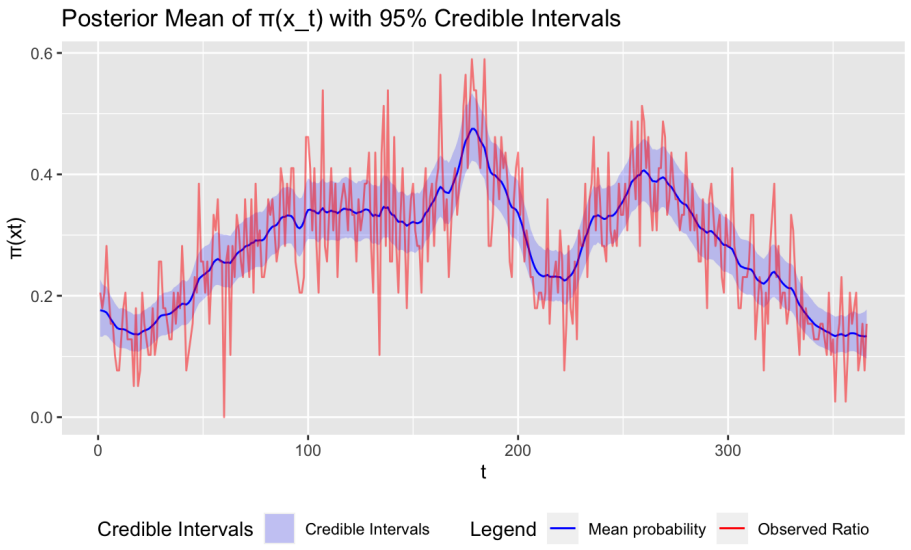


Figure 12: Posterior mean and credible intervals

Below is the code used to arrive at the above results.

```
# declaring parameters
alpha <- 2
beta <- 0.05
iters <- 50000
M <- 30

# loading data
load(file = "project2/data/rain.rda")
y <- rain[, "n.rain"]
n <- rain[, "n.years"]

# expit function
expit <- function(x) {1/(1+exp(-x))}

# create Q
get_Q <- function() {
  Q <- matrix(0, 366, 366)
  diag(Q) <- 2
  for (i in 1:365) {
```

```
    Q[i, i+1] <- -1
    Q[i+1, i] <- -1
  }
  Q[1,1] <- 1
  Q[366,366] <- 1
  return(Q)
}

mcmc_sampler <- function(alpha, beta, iters, M, burnin=ceiling(
    iters/10)){
  # declaring variables
  iters = iters + burnin
  n_chunks <- ceiling(366/M)

  # declaring initial data structures
  chain <- matrix(nrow=iters, ncol=367)
  acceptance_probs = matrix(nrow=iters-1, ncol=n_chunks)


  # calculating initial x-value
  x <- (y+1)/(n+1)
  pi <- expit(x)

  # sampling first sigma
  sigma_u_sq <- 1/rgamma(1, alpha, beta)

  # setting initial row for matrix
  chain[1,] <- c(x, sigma_u_sq)

  # start and end indices for all chunks
  a_idx <- seq(from = 1, to = floor(366 / M) * M + 1, by = M)
  b_idx <- a_idx+M-1
  b_idx[n_chunks] <- 366

  # create matrix Q
  Q <- get_Q()

  #lists to store precomputed matrices
  Q_AA_inv_matrices <- list()
  choleskys <- list()
  Q_AA_inv_Q_AB_matrices <- list()

  #precompute matrices
  for (ch in 1:n_chunks) {
    A <- a_idx[ch]:b_idx[ch]
    Q_AA <- Q[A, A]
    Q_AA_inv <- solve(Q[A,A])
    Q_AA_inv_matrices[[ch]] <- Q_AA_inv
    choleskys[[ch]] <- t(chol(Q_AA_inv))

    Q_AB <- Q[A, setdiff(1:366, A)]
    Q_AA_inv_Q_AB_matrices[[ch]] <- Q_AA_inv%*%Q_AB
  }

  # outer loop used for sampling x-vector
  for (iter in 2:iters){
    # uniform sampling used for accepting proposals
```

```
    u <- runif(n_chunks)

    # vectors for new samples and previous samples respectively
    new_x = c()
    old_x = chain[iter-1, ]

    # Gibb's sampling for sigma
    sigma_alpha <- alpha + 365/2
    sigma_beta <- beta + 1/2*sum((old_x[2:366] - old_x[1:365])^2)
    sigma_u_sq <- 1/rgamma(1, shape=sigma_alpha, scale=sigma_beta)

    # inner loop used for sampling x days
    for (ch in 1:n_chunks) {
      Q_AA_inv_Q_AB = Q_AA_inv_Q_AB_matrices[[ch]]

      # indexes able to handle all three cases mentioned in the
    problem
      a <- a_idx[ch]
      b <- b_idx[ch]
      x_b <- old_x[setdiff(1:366, a:b)]

      # calculates vectors and matrices needed for drawing
    conidtional multivariate normal
      x_b <- matrix(x_b, nrow = length(x_b), ncol = 1)
      mu <- -1*Q_AA_inv_Q_AB %*% x_b
      cholesky <- sqrt(sigma_u_sq)*choleskys[[ch]]

      # draws mulitvariate normal using cholesky decomposition
      proposal <- mu + cholesky%*%matrix(rnorm(b-a+1), nrow = b-a
    +1, ncol = 1)

      # calculating new and old pi
      new_pi <- expit(proposal)
      pi <- expit(old_x[a:b])

      # calculating acceptance probability (alpha)
      y_ab <- y[a:b]
      n_ab <- n[a:b]
      log_likelihood_ratio <- sum(y_ab*(log(new_pi) - log(pi)) + (n
    _ab-y_ab)*(log(1-new_pi)-log(1-pi)))
      acceptance_prob <- min(1, exp(log_likelihood_ratio))

      # accepting proposal based on acceptance probability
      if (u[ch] < acceptance_prob) {
        new_x <- c(new_x, t(proposal))
      } else {
        new_x <- c(new_x, old_x[a:b])
      }

      # used to track observed acceptance probabilities
      acceptance_probs[iter-1, ch] <- (u[ch] < acceptance_prob)
    }


    # updates chain and time data structures
    chain[iter, ] <- c(new_x, sigma_u_sq)
```

```
    if (iter %% 1000 == 0){print(iter/iters)}
  }

  # calculates mean acceptance probabilities
  acceptance_probs <- colMeans(acceptance_probs)

  # declares list to return
  result_list <- list(
    chain = chain,
    acceptance_probs = acceptance_probs
  )
  return (result_list)
}

# calls the function and measures the runtime
start_time <- proc.time()[3]
results <- mcmc_sampler(alpha, beta, iters, M)
end_time <- proc.time()[3]
total_time = end_time - start_time
print(total_time)
```

Listing 2: Code for 1g

The below code was used for calculating means and credible intervals and plotting for both **1f** and **1g**.

```
# sets variables for returned values
chain <- results$chain
acceptance_probs <- results$acceptance_probs

burnin = ceiling(iters/10)

# traceplots of x_1, x_201, x_366, and sigma
plot(expit(chain[,1]), type="l", ylab="pi(x_1)")
title("Traceplot of pi(x_1) (including burnin)")

plot(expit(chain[,201]), type="l", ylab="pi(x_201)")
title("Traceplot of pi(x_201) (including burnin)")

plot(expit(chain[,366]), type="l", ylab="pi(x_366)")
title("Traceplot of pi(x_366) (including burnin)")

plot(chain[,367], type="l", ylab="sigma_sq")
title("Traceplot of sigma_sq (including burnin)")

# plot of acceptance probabilities for all days
plot(acceptance_probs, type="l")
abline(h = mean(acceptance_probs), col = "red", lty = 2)
legend("bottomright",
       legend = c("acceptance probabilities", "mean"),
       col = c("black", "red"),
       lty = 1,
       bty = "n")
title("Acceptance probabilities for each day")
print("Mean acceptance prob:")
print(mean(acceptance_probs))

library(ggplot2)
```

```r
library(coda)

# calculates probabilities
prob_chain <- expit(chain[, 1:366])

# creates a mcmc-object of the chain (excluding sigma)
mcmc_chain <- as.mcmc(prob_chain)

# calculates confidence intervals for sigma
mcmc_sigma <- as.mcmc(chain[, 367])
conf_sigma <- summary(mcmc_sigma)$quantile
print(mean(chain[, 367]))
print(conf_sigma[c(1, 5)])


# calculates confidence intervals
conf_levels = summary(mcmc_chain)$quantile

# calculates mean values of each day
mean_pixt <- colMeans(mcmc_chain[, 1:366])
t_values <- 1:366

# creates a dataframe with time-values, mean x-values and
    confidence levels
df <- data.frame(t=t_values, mean=mean_pixt, lwr=conf_levels[,1],
    upr=conf_levels[,5])
# names(df) <- c("Day", "Mean", "Lower", "Upper")

# plots the mean x-values and confidence intervals with y-values
    from the dataset
ggplot(df, aes(x = t)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr, fill="Credible Intervals"
    ), alpha = 0.2) +
  geom_line(aes(y = mean_pixt, color = "Mean probability")) +
  geom_line(aes(y = y / n, color = "Observed Ratio"), alpha=0.5) +
  labs(title = "Posterior Mean of pi(x_t) with 95% Credible
    Intervals",
       x = "t", y = "pi(xt)", color = "Legend") +
  scale_color_manual(values = c("Mean probability" = "blue", "
    Observed Ratio" = "red")) +
  scale_fill_manual(name = "Credible Intervals", values = "blue") +
      # Added for ribbon fill color
  theme(legend.position = "bottom")


# plots histograms of x_1, x_201, x_366 and sigma_sq
hist(expit(chain[burnin + 1:iters, 1]), main="Histogram of pi(x_1)
    (excluding burnin)", xlab="pi(x_1)", breaks=50)
hist(expit(chain[burnin + 1:iters, 201]), main="Histogram of pi(x_
    201) (excluding burnin)", xlab="pi(x_201)", breaks=50)
hist(expit(chain[burnin + 1:iters, 366]), main="Histogram of pi(x_
    366) (excluding burnin)", xlab="pi(x_366)", breaks=50)
hist(chain[burnin + 1:iters, 367], main="Histogram of sigma_sq (
    excluding burnin)", xlab="sigma_sq", breaks=50)


# plots autocorrelation of x_1, x_201, x_366 and sigma_sq
```

```
acf(expit(chain[burnin + 1:iters, 1]), main="Autocorrelation of pi(
    x_1) (excluding burnin)")
acf(expit(chain[burnin + 1:iters, 201]), main="Autocorrelation of
    pi(x_201) (excluding burnin)")
acf(expit(chain[burnin + 1:iters, 366]), main="Autocorrelation of
    pi(x_366) (excluding burnin)")
acf(chain[burnin + 1:iters, 367], main="Autocorrelation of sigma_sq
    (excluding burnin)")
```

Listing 3: Code for plotting

2 INLA

a)

The predictions from fitting the model are shown in figure 13. We observe qualitatively that the predictions are very similar to the results from part 1.

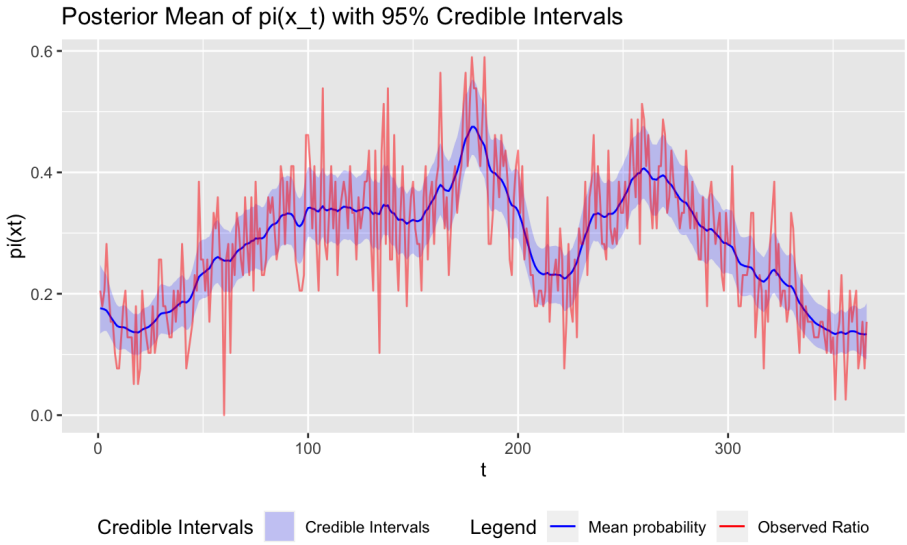


Figure 13: Posterior mean parameter values given by INLA

In order to enforce the same prior as in part 1f), we set prior = "loggamma", param = c(2, 0.05). We need to set use a logarithmic gamma density, since the prior in the INLA library is defined on $\theta = \log(\frac{1}{\sigma_u^2})$. Looking at figure 14 we can see that the INLA estimates are largely the same as the MCMC estimates. The mean difference in posterior mean is 0.0003 which is very small.

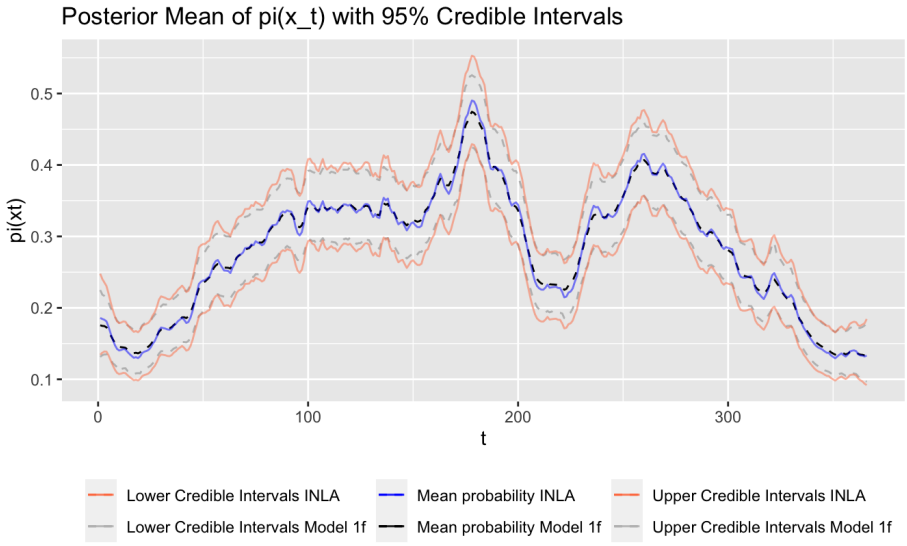


Figure 14: Comparison of INLA and model from **1f**

The expected precision is estimated to be 106.7317, slightly lower than the expected value of $1/\sigma_u^2$ found in **1f**. It took approximately 2.9 seconds to run the INLA-model, which is singificantly faster than the MCMC-model from **1f** and **1g**.

b)

We assess how sensitive the model is to the control parameters by comparing to a model fitted with the Gaussian strategy. The models yield very similar results (however the Gaussian approximation is consistently slightly faster). Figure 15 shows the model with Gaussian strategy while figure 16 shows the comparison between the model used in **2a** (simplified.laplace strategy) and the model with gaussian strategy. The lines for the model using Gaussian strategy are opaque, and the lines for the model from **a** are dashed. There is no noticeable difference between the models as the mean difference between the posterior means are $2.38E-12$.

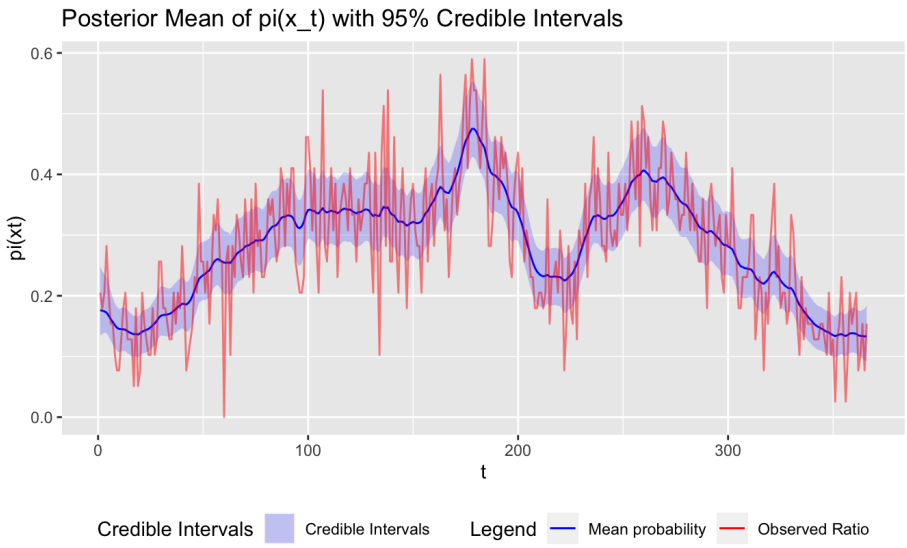


Figure 15: Inla predictions with Gaussian strategy

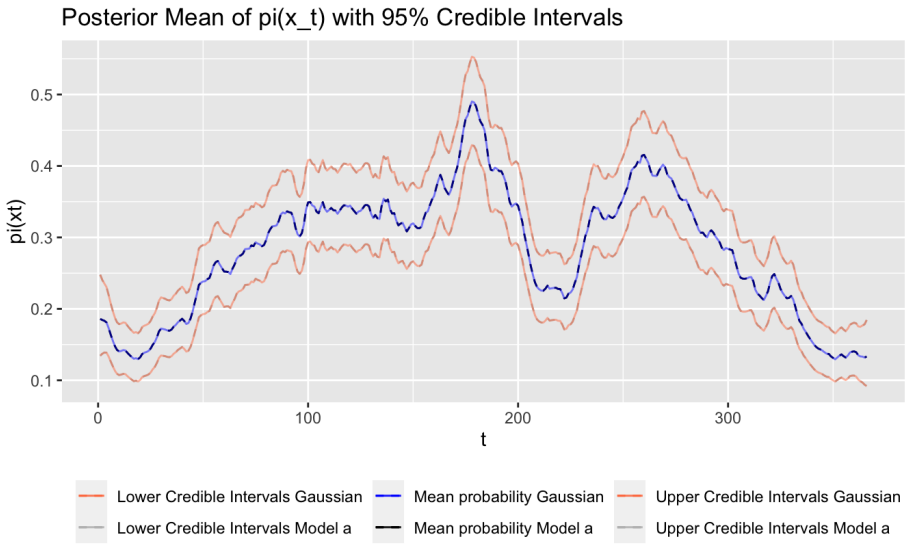


Figure 16: First model compared to model with Gaussian strategy

The other control input used is the int.strategy. In **2a** the strategy "ccd" is used. We compare the results to the "grid"-strategy. From figure 17 the plot seems very similar to the original strategy. In figure 18 there is no noticeable difference. The mean difference between using "grid"-strategy and "ccd" is

$2.88E-6$ which is significantly higher than with gaussian strategy, but still very low. Therefore we conclude that the model is very robust to the input control parameters.

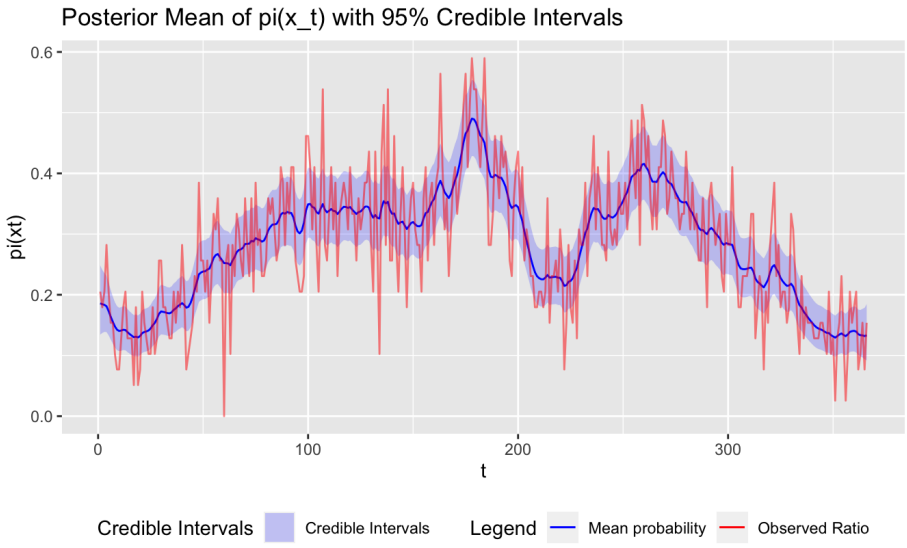


Figure 17: Inla predictions with grid strategy

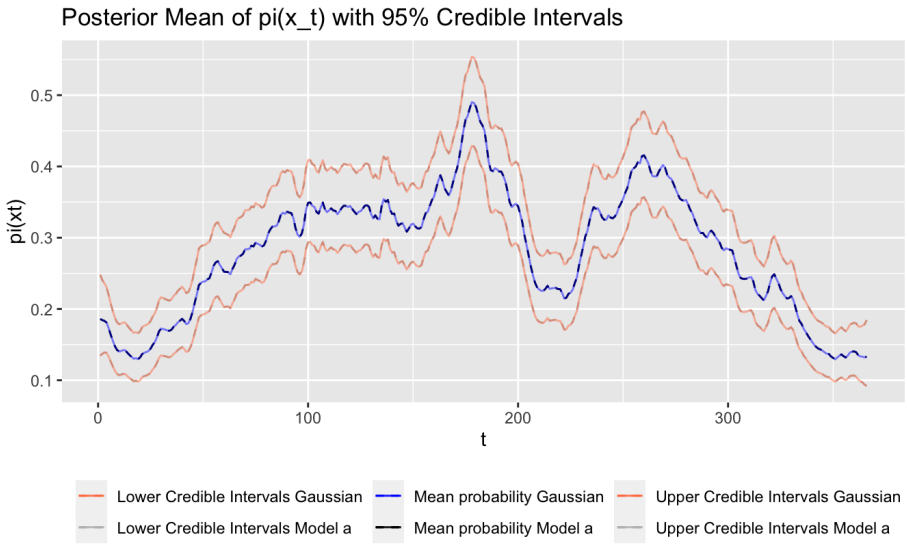


Figure 18: First model compared to model with grid strategy

c)

The only difference from the model in a) is that we include an intercept in the random walk model, in addition to the random effect. The original random walk is formulated as such:

$$x_t = x_{t-1} + u_t \Rightarrow x_t = x_1 + \sum_{j=2}^{t} u_j \tag{1}$$

While the new model is:

$$x_t = \lambda + x_1 + \sum_{j=2}^{t} u_j \tag{2}$$

Where lambda is the intercept with the default non-informative (improper) prior $\lambda \sim N(0, \tau^{-1}), \tau = 0$. We do not expect this to affect the results of the model, as the model density on the other parameters is unchanged when shifting by the intercept, since

$$\prod_{j=2}^{366} \frac{1}{\sigma_u} \exp\left\{ -\frac{1}{2\sigma_u^2} (x_j - x_{j-1})^2 \right\} = \prod_{j=2}^{366} \frac{1}{\sigma_u} \exp\left\{ -\frac{1}{2\sigma_u^2} ((x_j + \lambda) - (x_{j-1} + \lambda))^2 \right\}$$

The addition of the intercept also does not change the prior on x_1 since the prior on the intercept in non-informative. The predictions are indeed very similar, as shown in figures 19 and 20 that show plots of the INLA predictions without the intercept, and with the intercept respectively. The mean and sd estimates for $\frac{1}{\sigma_u^2}$ are 106.735 and 27.569, which is very similar to part a).

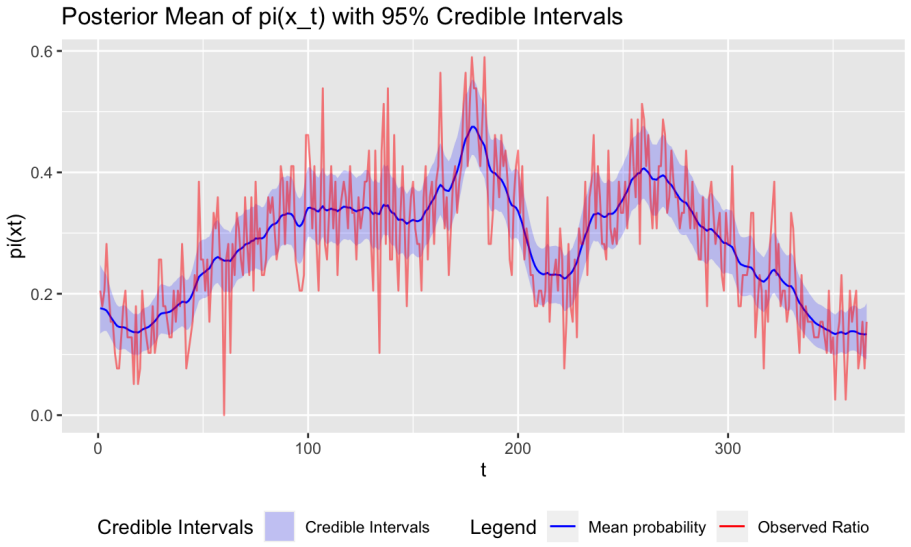


Figure 19: Inla predictions without intercept

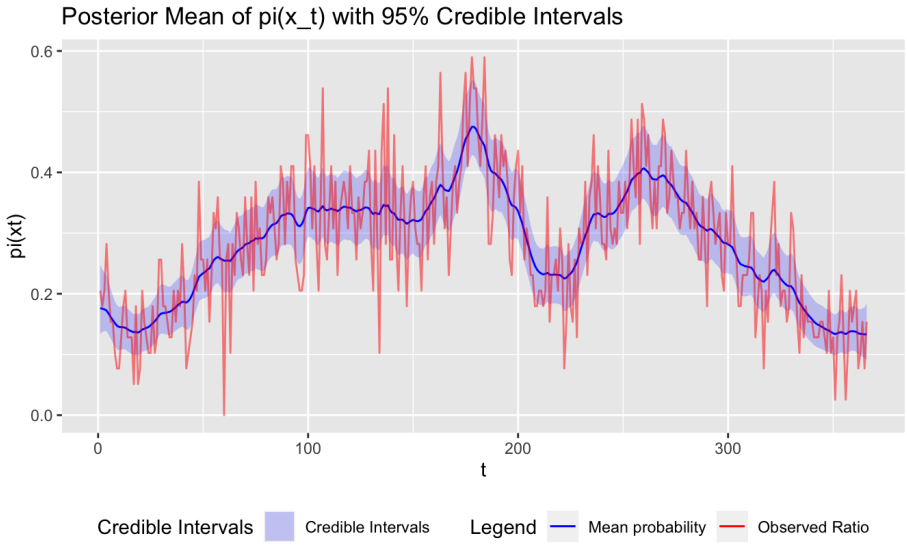


Figure 20: Inla predictions with intercept

The difference between the models are the values of the random effects x_t. They need to change to maintain the same values of π due to the presence of λ in the second model. The predictions are not different since you can use both models in equations 1 and 2 to arrive at the best fit for π. Figure 21 shows the posterior density of the intercept. Figures 22 and 23 show the random effects without the intercept, and with it respectively. We can see that the plots are nearly identical if you subtract the expected intercept value -0.988 from the random effects of the model with intercept (note the differing values on the vertical axes).

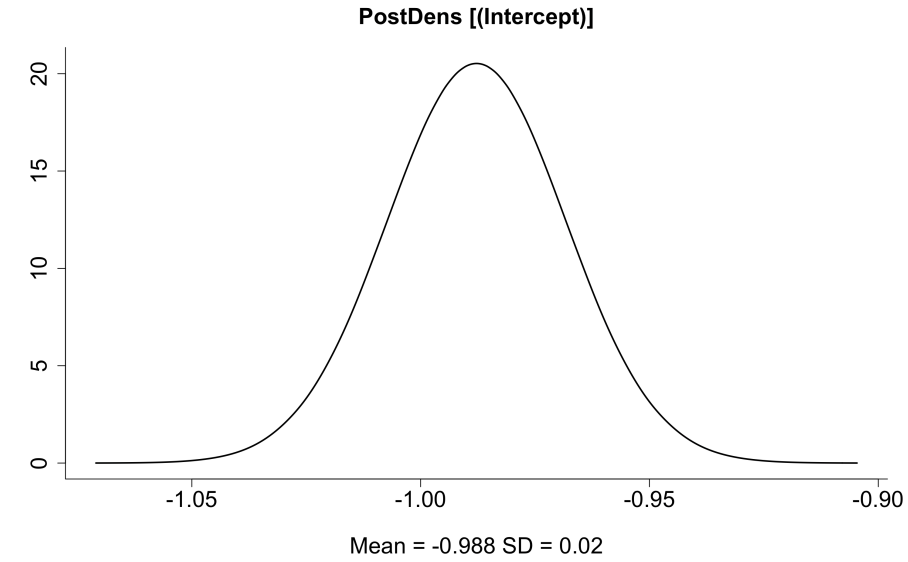


Figure 21: The posterior intercept distribution

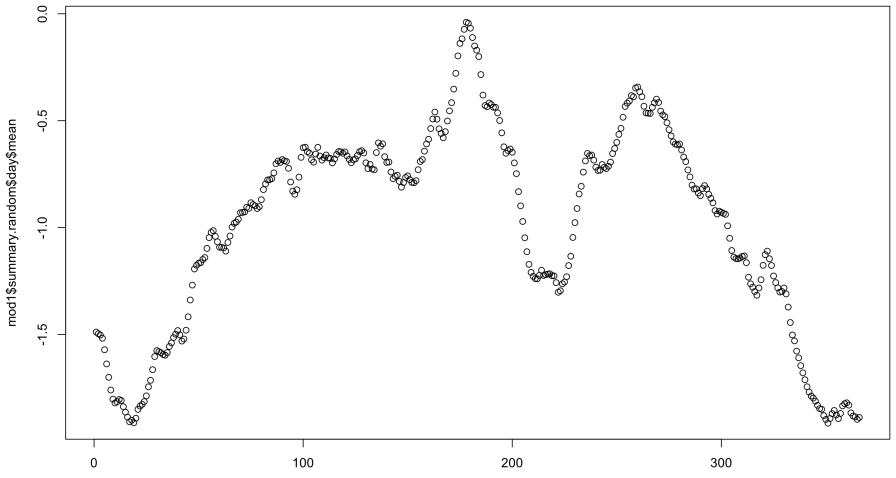


Figure 22: Random effects without intercept

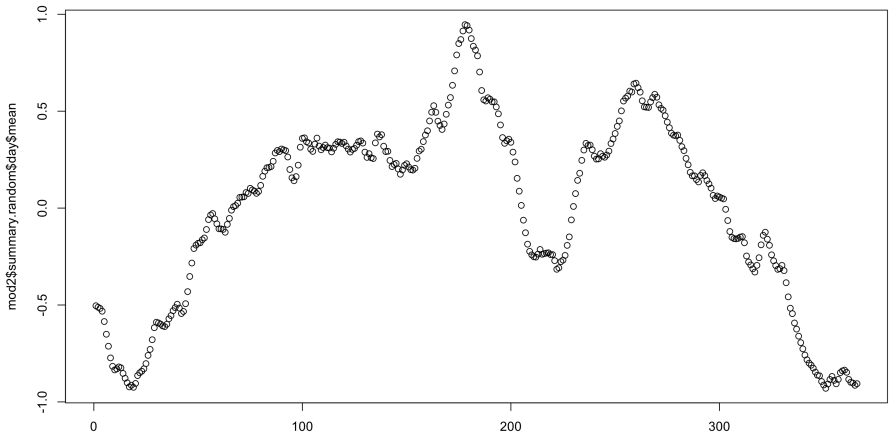


Figure 23: Random effects with intercept

The following is the code for problem 2:

```
# Import the INLA library
library("INLA")

# Load the rain data file
load(file="./data/rain.rda")

# Initialize the timer for timing the computation
ptm <- proc.time()

# Set the control parameters
control_inla <- list(strategy = "simplified.laplace", int.strategy
    = "ccd")

# Declare the formula for part a)
formula1 = n.rain ~ -1 + f(
    day,
    model = "rw1",
    constr = FALSE,
    hyper = list(prec = list(prior = "loggamma", param = c(2, 0.05)
    )))

# Fit the inla model on formula1
mod1 <- inla(formula1,
             data=rain,
             Ntrials=n.years,
             control.compute=list(config = TRUE),
             family="binomial",
             control.inla=control_inla)

# Print the elapsed time
print("ELAPSED TIME")
print(proc.time() - ptm)

# Declare the formula for part c)
```

```r
formula2 = n.rain ~ f(
    day,
    model="rw1",
    constr=TRUE,
    hyper = list(prec = list(prior = "loggamma", param = c(2, 0.05)
    )))

# Fit the inla model on formula2
mod2 <- inla(formula2,
             data=rain,
             Ntrials=n.years,
             control.compute=list(config = TRUE),
             family="binomial",
             control.inla=control_inla)
```

Listing 4: Code for INLA Model

3 RTMB

We start by computing the negative log joint likelihood:

$$
\begin{aligned}
& -\log(p(y,x|\theta)) \\
=& -\log(p(y|X,\sigma_u^2)p(x|\sigma_u^2)) \\
=& -\log(p(y|X)p(x|\sigma_u^2)) \\
=& -\log(\prod_{t=1}^{T}\binom{n_t}{y_t}\pi(x_t)^{y_t}(1-\pi(x_t)^{n_t-y_t})) - \log(\prod_{t=2}^{T}\frac{1}{\sigma_u}e^{-\frac{1}{2\sigma_u^2}(x_t-x_{t-1})^2}) \\
=& -(T-1)\log(\frac{1}{\sigma_u}) - \sum_{t=1}^{T}\log(\binom{n_t}{y_t}) - \sum_{t=1}^{T}y_t\log(\pi(x_t)) - \sum_{t=1}^{T}(n_t-y_t)\log(1-\pi(x_t)) + \frac{1}{2\sigma_u^2}\sum_{t=2}^{T}(x_t-x_{t-1})^2
\end{aligned}
$$

Since we are looking for the argmin of σ_u and x_t, we can remove the constants and minimize:

$$
(T-1)\log(\sigma_u) - \sum_{t=1}^{T}y_t\log(\pi(x_t)) - \sum_{t=1}^{T}(n_t-y_t)\log(1-\pi(x_t)) + \frac{1}{2\sigma_u^2}\sum_{t=2}^{T}(x_t-x_{t-1})^2
$$

The RTMB model yields largely the same results as the other two. Figure 24 shows the RTMB estimates while Figure 25 displays all the models in the same plot for comparison. Credible intervals are not included in this plot to improve visiblity.

The σ^2 is estimated to be 0.007, slightly higher than the σ^2 found in **1f** and slightly lower than what was found in **2a**. The corresponding estimated precision is 143.65, higher than the precision in 2, and lower than the one in 1. It took approximately 1.44 seconds to run the RTMB-model.

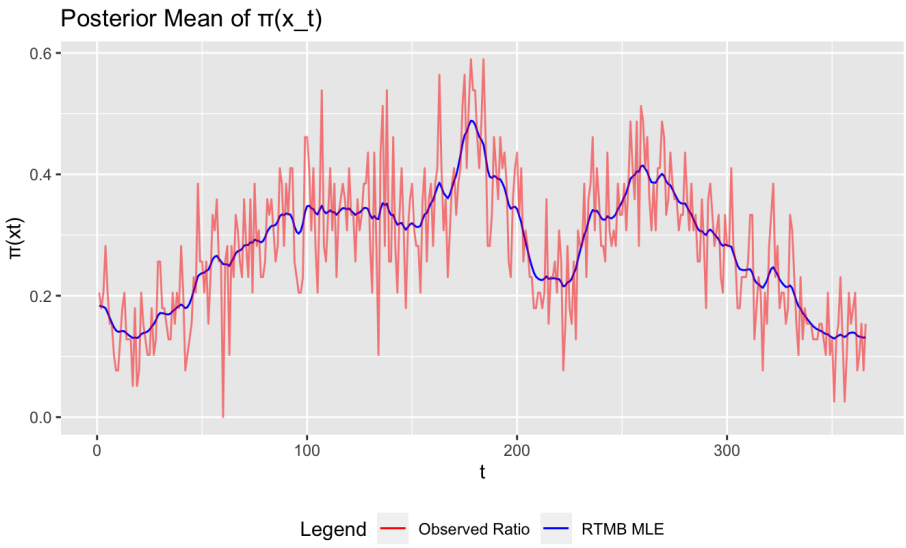


Figure 24

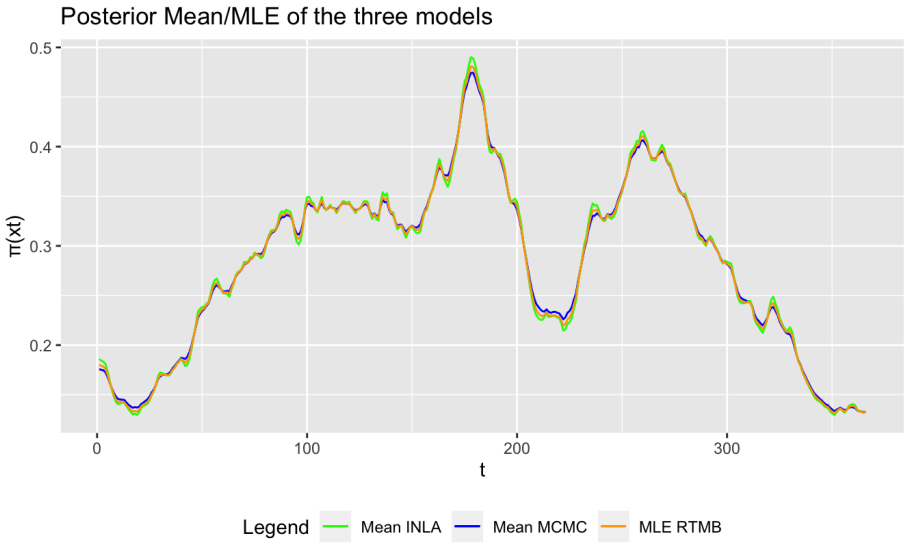


Figure 25

Below is the code to compute the maximum likelihood estimates with RTMB:

```
# Import RTMB
```

```r
library("RTMB")

# Load the rain data
load(file = "project2/data/rain.rda")

# Define the expit function
sigmoid <- function(x) {
  return(1 / (1 + exp(-x)))
}

# logit function
logit <- function(x) {1/(1+exp(-x))}

# Define the negative log likelihood
f <- function(p) {

  y <- rain$n.rain
  n <- rain$n.years

  # Compute the binomial terms of the log likelihood
  bin_terms <- -sum(y * log(sigmoid(p$x))) -
    sum((n - y) * log(1 - sigmoid(p$x)))

  # Compute the random walk terms of the log likelihood
  rw_terms <- (length(p$x) - 1) * log(p$var ** (1 / 2)) +
    (1 / (2 * p$var)) * sum((p$x[-1] - p$x[-length(p$x)])**2)

  return(bin_terms + rw_terms)
}

# Set the random effects to 0
x <- rain$n.rain * 0
# Set the variance
var <- 0.05

# Invoke the autodifferentiator
start_time <- proc.time()[3]
obj <- MakeADFun(f, list(x = x, var = var), random="x",silent =
    TRUE)
# Obtain the maximum likelihood estimates
opt <- nlminb(obj$par, obj$fn, obj$gr)

end_time <- proc.time()[3]
total_time = end_time - start_time
print(total_time)

parameters <- sdreport(obj)
```

Listing 5: Code for RTMB Model