STA-3001 Assignment 4 - MCMC & INLA

Harald Karlsen

Spring 2026

1 Problem 1

1.1 Task 1a)

We will use a uniform proposal density centered at $\theta^{(t)}$:

$$q(\theta^*|\theta^{(t)}) = \frac{1}{2d}, \qquad \theta^{(t)} - d < \theta^* < \theta^{(t)} + d \tag{1}$$

with some chosen interval length d. Note that since $\theta \in (0, 1)$, we will truncate θ^*:

$$\theta^* \sim Unif(\max(0, \theta^{(t)} - d), \min(1, \theta^{(t)} + d)). \tag{2}$$

The acceptance probability will then simplify to,

$$\alpha(\theta^*|\theta^{(t)}) = \min\left\{1, \frac{\pi(\theta^*|\mathbf{y})}{\pi(\theta^{(t)}|\mathbf{y})}\right\}, \tag{3}$$

where $\pi(\theta|\mathbf{y})$ is the target density. Using Bayes' law we get,

$$\alpha(\theta^*|\theta^{(t)}) = \min\left\{1, \frac{\theta^{*(\alpha-1)}(1-\theta^*)^{\beta-1}\prod_{i=1}^n f(y_i|\theta^*)}{\theta^{(t)(\alpha-1)}(1-\theta^{(t)})^{\beta-1}\prod_{i=1}^n f(y_i|\theta^{(t)})}\right\}, \tag{4}$$

(posterior $\propto$ likelihood $\times$ prior) when inserting the prior of θ and the likelihood function of the sample.

The goal with the Metropolis-Hastings algorithm is to generate a Markov Chain $\{\theta^{(t)}\}_{t=0}^T$ with $\pi(\theta|\mathbf{y})$ as its stationary distribution. We will start with an initial value $\theta^{(0)} = 0.5$ (middle of interval). Then, for every time-step $t = 0, 1, ..., T-1$ we draw a candidate θ^* from $q(\theta^*|\theta^{(t)})$. We then compute the acceptance probability using equation 4, and draw $u \sim Unif(0, 1)$. If $u \leq \alpha(\theta^*|\theta^{(t)})$ then we update the state to θ^*, otherwise the state stays the same:

$$\theta^{(t+1)} = \begin{cases} \theta^*, & u \leq \alpha(\theta^*|\theta^{(t)}) \\ \theta^{(t)}, & u > \alpha(\theta^*|\theta^{(t)}) \end{cases} \tag{5}$$

1.2 Task 1b)

We start by generating the random data from the mixture model using the following code:

```r
set.seed(67)
theta.true <- 0.4
n <- 1000
mu1 <- 0
sig1 <- 1
mu2 <- 3
sig2 <- 1.5
alpha <- 1
beta <- 1
gauss1 <- rnorm(n, mu1, sig1)
gauss2 <- rnorm(n, mu2, sig2)
#Bernoulli variable determining which gaussian we draw from
z <- rbinom(n, size=1, prob=theta.true)
#random sample of y
y <- ifelse(z==1,gauss1,gauss2)
```

To ensure numerical stability we will work on the log-scale. We implement the log-likelihood and log-prior with the following code:

```r
log_prior <- function(theta, alpha, beta) {
  val <- dbeta(theta, alpha, beta, log=TRUE)
  return(val)
}

log_likelihood <- function(theta, y) {
  f <- theta*dnorm(y,mu1,sig1)+(1-theta)*dnorm(y,mu2,sig2)
  if (any(f<=0)) return(-INF)
  L <- sum(log(f))
  return(L)
}
```

These functions are used to calculate the acceptance probability in our Metropolis-Hastings algorithm:

```r
metropolis_hastings <- function(y, T, d, alpha, beta,
↪  theta.init=0.5) {
  theta <- numeric(T)
  accept_prob <- numeric(T)
  accept_tracker <- numeric(T)
  theta[1] <- theta.init

  for (t in 1:(T-1)) {
    theta.curr <- theta[t]
    lb <- max(0, theta.curr-d)
    ub <- min(1, theta.curr+d)
    theta.prop <- runif(1,lb,ub)

    #Computing prior and likelihood terms with current and proposal
    ↪  theta
    prior.curr <- log_prior(theta.curr, alpha, beta)
    prior.prop <- log_prior(theta.prop, alpha, beta)
```

```
    likelihood.curr <- log_likelihood(theta.curr, y)
    likelihood.prop <- log_likelihood(theta.prop, y)

    #Computing log posteriors (up to a constant)
    posterior.curr <- prior.curr+likelihood.curr
    posterior.prop <- prior.prop+likelihood.prop

    #Computing acceptance probability
    accept_prob[t] <- min(1, exp(posterior.prop-posterior.curr))

    #Sampling from uniform distribution to accept/reject
    u <- runif(1,0,1)
    #Reject if u > a
    if (u > accept_prob[t]) {
      accept_tracker[t] <- 0
      theta[t+1] <- theta.curr
    } else {
      accept_tracker[t] <- 1
      theta[t+1] <- theta.prop
    }
  }
  list(theta=theta, accept_prob = accept_prob, accepted =
  ↪  accept_tracker)
}
```

We initialize the algorithm with $\theta^{(0)} = 0.5$, and use an update range of $d = 0.1$. We tried a few different d values and chose the one that resulted in the best ACF plot. Note that since there are no given values for α and β in the prior distribution of θ, we just assume both of them are 1. In figure 1 we see the trace plot of the θ values obtained by the Metropolis-Hastings algorithm. As we can see, all the θ values after the initial step stay in the range $(0.30, 0.45)$ and the plot is most dense around ~ 0.37.

In the ACF plot (figure 2) we see that the autocorrelation decays to ~ 0 at around lag 8, meaning that samples are essentially independent after 8 iterations of the algorithm. This suggests that the Markov Chain has converged to its stationary distribution.

In figure 3 we see the acceptance rate as a function of iterations. As we can see, most of the values are around 0, which is expected since we have chosen quite a large value for d in order to let the algorithm explore.

1.3 Task 1c)

To get a posterior point estimate we simply take the average value of the Markov Chain samples of θ, which gave the estimate $\hat{\theta} = 0.372215$. This is a decent result given that we only have 1000 observations. The 95% credible interval (equi-tailed) is $(0.334, 0.409)$, which does indeed contain the true θ value. In figure 4 we see the histogram of the sampled Markov chain. Note that for the point estimate and CI we used a burn-in period of 2000 samples.

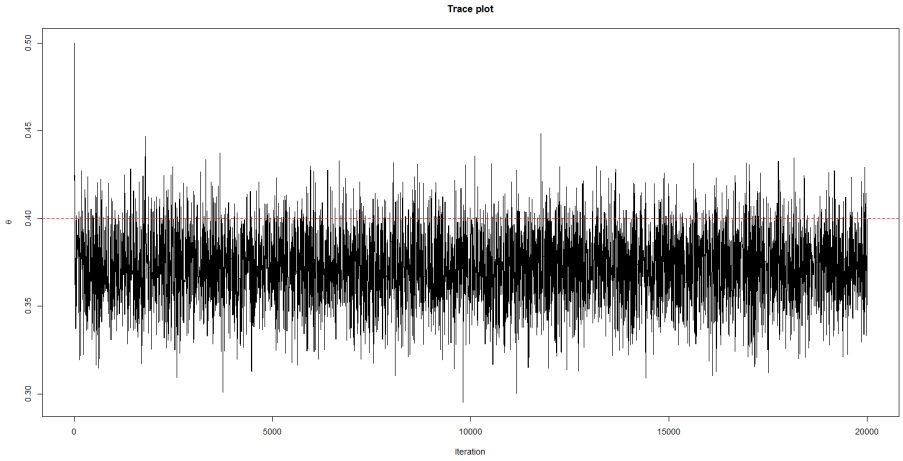


Figure 1: Trace plot of θ-values using Metropolis-Hastings.

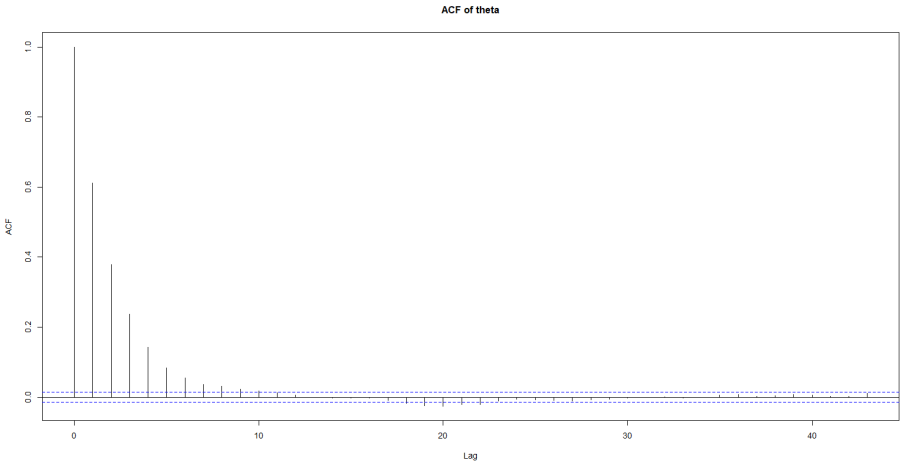


Figure 2: Autocorrelation function for sampled Markov chain.

```r
#Posterior point estimate
theta.burnin <- theta.sampled[2000:length(theta.sampled)]
theta.hat <- mean(theta.burnin)
#95% credible interval
credible_interval <- quantile(theta.burnin, c(0.025, 0.975))
```

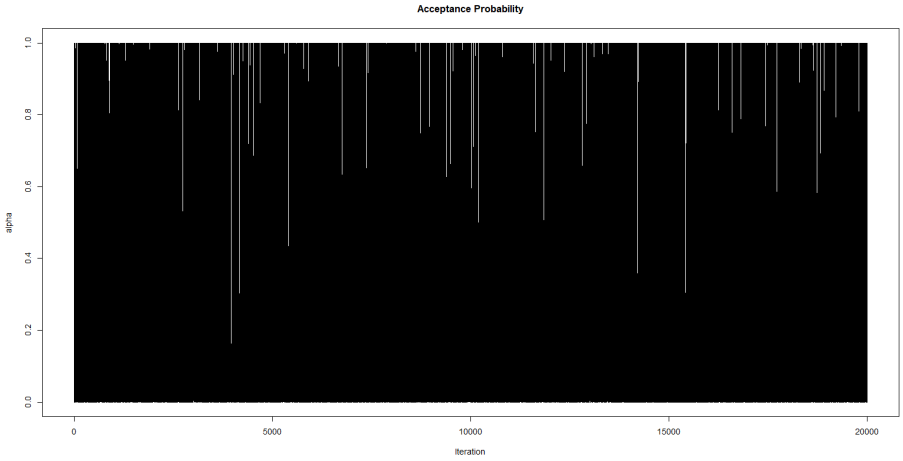


Figure 3: Acceptance rate as a function of iterations.

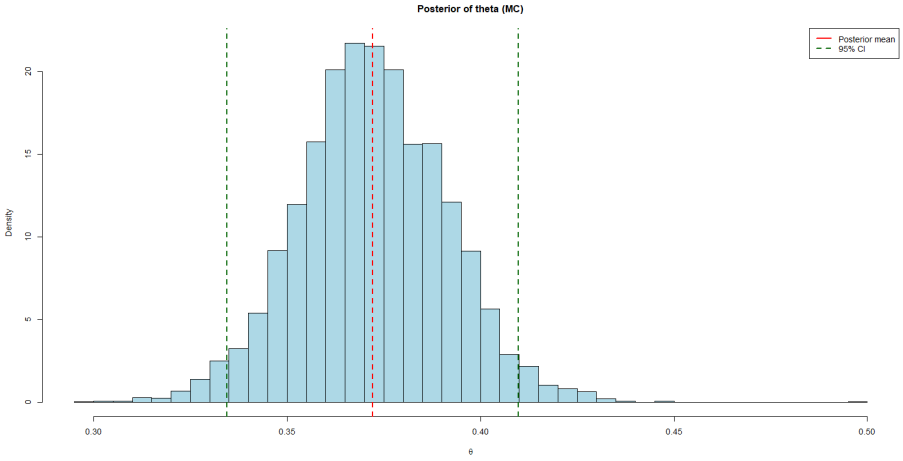


Figure 4: Histogram for sampled Markov chain.

2 Problem 2

2.1 Task 2a)

The priors of α and β are on the form:

$$\pi(\alpha) \propto \exp\left(-\frac{\tau}{2}\alpha^2\right)$$
$$\pi(\beta) \propto \exp\left(-\frac{\tau}{2}\beta^2\right)$$

The conditional distribution for a sample y_i, $\pi(y_i|\alpha,\beta)$ will be $Binomial(n_i, p_i)$ since each of the n_i beetles have a probability p_i of being dead:

$$\pi(y_i|\alpha,\beta) = \binom{n_i}{y_i} p_i^{y_i}(1-p_i)^{n_i-y_i}. \tag{6}$$

Thus, using Bayes law we get that the conditional posteriors of α and β are:

$$\pi(\alpha|\mathbf{y},\beta) \propto \exp\left(-\frac{\tau}{2}\alpha^2\right)\prod_{i=1}^{8}\binom{n_i}{y_i}p_i^{y_i}(1-p_i)^{n_i-y_i} \tag{7}$$

$$\pi(\beta|\mathbf{y},\alpha) \propto \exp\left(-\frac{\tau}{2}\beta^2\right)\prod_{i=1}^{8}\binom{n_i}{y_i}p_i^{y_i}(1-p_i)^{n_i-y_i} \tag{8}$$

Note that since the proposals are symmetric Gaussian random walks, the proposal terms will cancel in the acceptance probability and leave only the posterior:

$$\alpha_\alpha(\alpha^*, \alpha^{(t)}) = \min\left\{1, \frac{\pi(\alpha^*|y,\beta^{(t)})}{\pi(\alpha^{(t)}|y,\beta^{(t)})}\right\} \tag{9}$$

$$\alpha_\beta(\beta^*, \beta^{(t)}) = \min\left\{1, \frac{\pi(\beta^*|y,\alpha^{(t+1)})}{\pi(\beta^{(t)}|y,\alpha^{(t+1)})}\right\} \tag{10}$$

The standard deviations of the Gaussian random walks will control the scale of the proposals. A small σ value will keep the proposals close the current value, which then leads to slow convergence, high autocorrelation and a high acceptance rate. A large σ value lets the algorithm explore a larger region, which makes proposals appear in low posterior density regions, giving a low acceptance rate. Ideally we want something in the middle.

2.2 Task 2b)

We implement the algorithm using the following code:

```
beta_posterior <- function(alpha, beta, x, y, n, tau) {
  #Computing mortality rate
  eta <- alpha + beta*x
  #Inverse logit to compute p
  p <- 1 / (1+exp(-eta))
  #Prior
  prior <- dnorm(beta,0,1/sqrt(tau))
  #Likelihood
  likelihood <- prod(dbinom(y,n,p))
  return(prior*likelihood)
}

mh_gibbs <- function(x, y, n, tau, T, sig.a, sig.b, alpha0=0,
  ↪  beta0=0) {
  alpha.sampled <- numeric(T)
```

```r
beta.sampled <- numeric(T)
alpha.accept <- numeric(T)
beta.accept <- numeric(T)

alpha.sampled[1] <- alpha0
beta.sampled[1] <- beta0

for (t in 1:(T-1)) {
  alpha.curr <- alpha.sampled[t]
  beta.curr <- beta.sampled[t]

  #Updating alpha using current beta
  #Drawing random proposal for alpha
  alpha.prop <- rnorm(1, alpha.curr, sig.a)
  #computing posteriors
  alpha.prop.post <- alpha_posterior(alpha.prop, beta.curr, x, y, n, tau)
  alpha.curr.post <- alpha_posterior(alpha.curr, beta.curr, x, y, n, tau)
  #Computing acceptance probability
  alpha.accprob <- min(1, alpha.prop.post / alpha.curr.post)
  #Determining if we accept/reject proposal
  u.alpha <- runif(1,0,1)
  if (u.alpha > alpha.accprob) {
    alpha.sampled[t+1] <- alpha.curr
    alpha.accept[t] <- 0
  } else {
    alpha.accept[t] <- 1
    alpha.sampled[t+1] <- alpha.prop
  }
  #Using the updated alpha to update beta
  alpha.curr <- alpha.sampled[t+1]
  #Drawing random proposal for beta
  beta.prop <- rnorm(1, beta.curr, sig.b)
  #computing posteriors
  beta.prop.post <- beta_posterior(alpha.curr, beta.prop, x, y, n, tau)
  beta.curr.post <- beta_posterior(alpha.curr, beta.curr, x, y, n, tau)
  #Computing acceptance probability
  beta.accprob <- min(1, beta.prop.post / beta.curr.post)
  #Determining if we accept/reject proposal
  u.beta <- runif(1,0,1)
  if (u.beta > beta.accprob) {
    beta.sampled[t+1] <- beta.curr
    beta.accept[t] <- 0
  } else {
    beta.accept[t] <- 1
    beta.sampled[t+1] <- beta.prop
  }
```

```
  }
  list(alpha=alpha.sampled, beta=beta.sampled,
  ↪  alpha_accept=alpha.accept, beta_accept=beta.accept)
}
```

We start with standard deviation values of 0.5 for the proposal densities. For α this gave an acceptance rate of 0.3014, which is within the $0.2 - 0.5$ range that is considered good for random walk schemes. For β we got an acceptance rate of $\sim$ 94% which suggests we need to increase σ_β. In table 1 we see an overview of all the values we tested, and we eventually landed on $\sigma_\beta = 6$. Note that when we increased σ_β to 6 the acceptance rate of α when up to 31.45%.

In figure 5 we see the trace plots for the two parameters. The patterns of the trace plots suggests that we have chosen good values for σ_α and σ_β. The α values are clumped around $\alpha \approx 0.75$, while the β values are around $\beta \approx 33.86$, which we could use as posterior point estimates.

In figure 6 we see the ACF plots for α and β. The ACF goes to zero at around lag 11 for α and lag 10 for β, which suggests that the Markov chain will eventually converge to its stationary distribution. Figure 7 show the acceptance probability plots, which just confirm what we already mentioned about the impact of σ_α and σ_β.

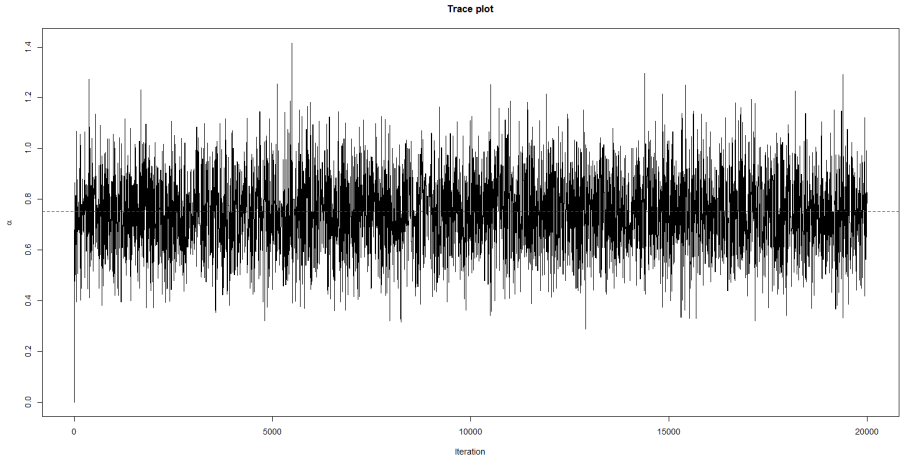
σ_β	Acceptance rate (%)
0.5	94%
1	88%
2	78%
3	68%
5	53%
6	48%

Table 1: Acceptance rate for various choices of σ_b.

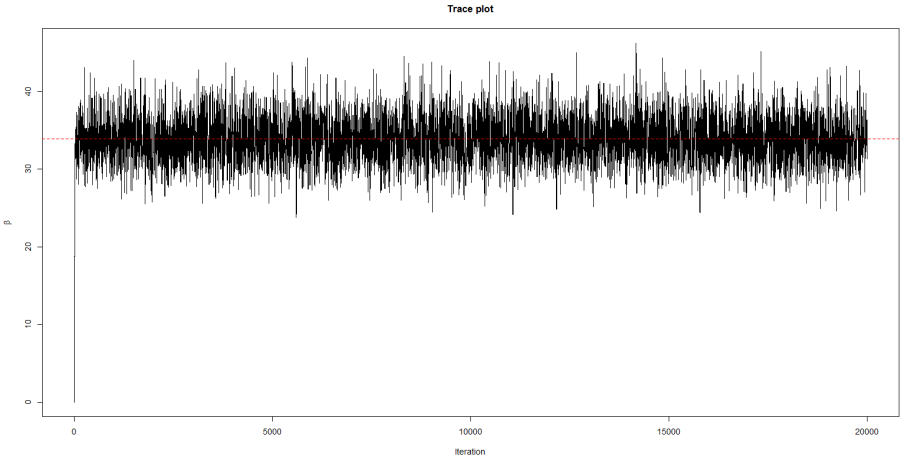
2.3 Task 2c)

The INLA implementation:

```
formula = Number_killed ~ x
result = inla(formula, data = beetle,
              family = "binomial", Ntrials = Number_tested)
#Extracting info from INLA summary
inla.marginals <- result\$marginals.fixed
alpha.marginal <- inla.marginals\$'(Intercept)'
beta.marginal <- inla.marginals\$x
#Computing point estimates
alpha.mean <- inla.emarginal(function(x) x, alpha.marginal)
beta.mean <- inla.emarginal(function(x) x, beta.marginal)
#Computing 95%-CIs
```



(a) Trace plot for α.

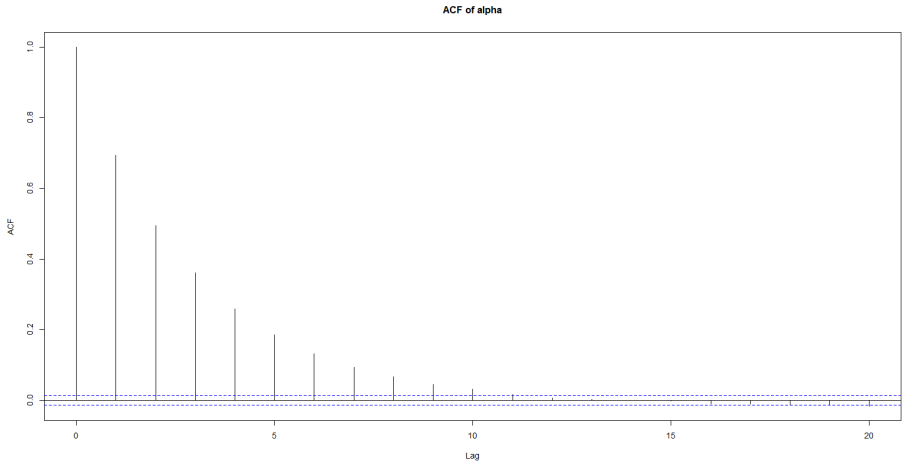


(b) Trace plot for β

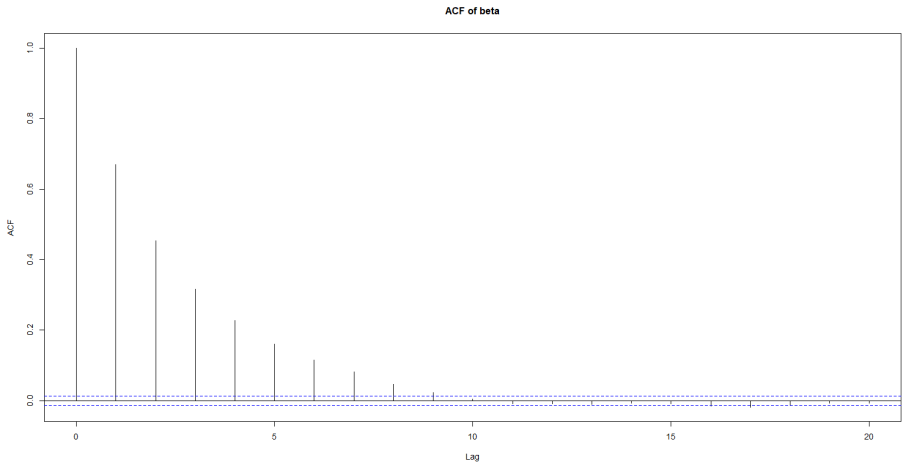
Figure 5: Trace plots for parameters.

```
12 alpha.ci.inla <- inla.qmarginal(c(0.025, 0.975), alpha.marginal)
```

The benefit of INLA is that it gives results of the same caliber as MCMC methods, while being easier to implement and at a smaller computational cost. Therefore we are interested in checking how close the INLA results are to the MCMC results. Table 2 shown the credible intervals and point estimates of the parameters for both of the algorithms. As we can see, the point estimate difference was *only* 0.0041 for α and 0.0343 for β. In figure 8 we see that the overlap of the marginal posterior densities are pretty good between the INLA



(a) ACF plot for α.



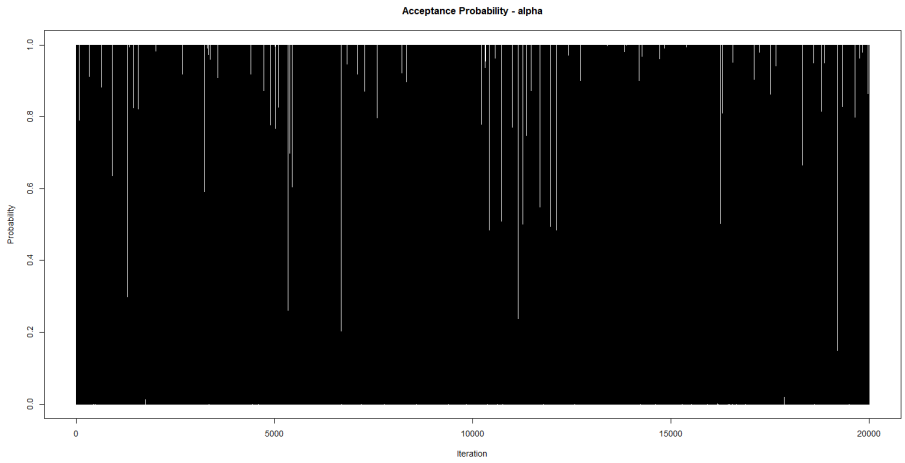
(b) ACF plot for β

Figure 6: ACF plots for parameters.

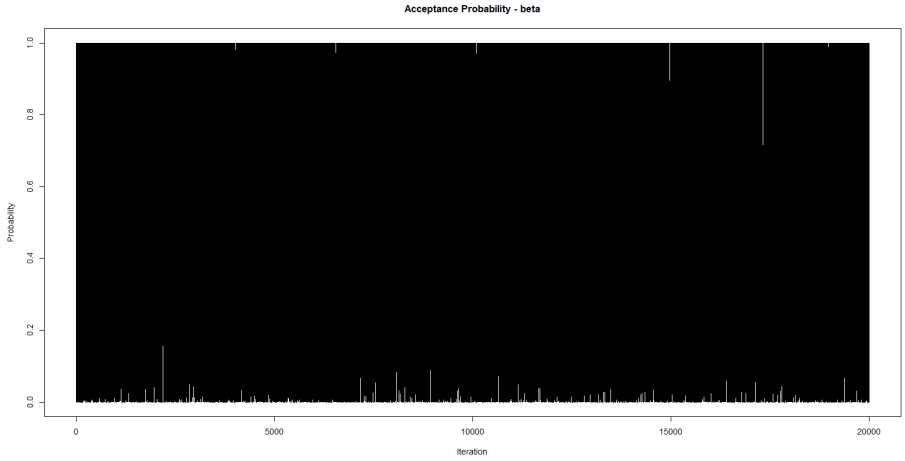
and MCMC methods, for both of the parameters.

2.4 Task 2d)

In figure 9 we have the estimated curves that we get when using the estimates for α and β from MCMC and INLA. As we can see, the curves overlap completely and fit reasonably well with the observed data points.



(a) Acceptance probability as a function of iterations for α.



(b) Acceptance probability as a function of iterations for β

Figure 7: Acceptance probability plots for parameters.

Parameter (Method)	95% Credible Interval	Point Estimate	Δ
α (MCMC)	$(0.479, 1.1018)$	0.7457	-
α (INLA)	$(0.482, 1.017)$	0.7498	0.0041
β (MCMC)	$(28.4, 39.8)$	33.9005	-
β (INLA)	$(28.2, 39.5)$	33.8662	0.0343

Table 2: Credible intervals, point estimates and difference between point estimates of MCMC and INLA.

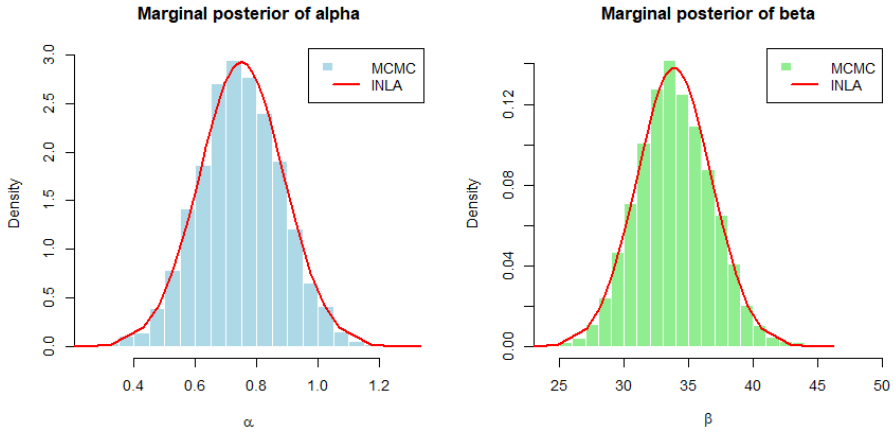


Figure 8: Marginal posterior densities for α and β with INLA and MCMC.

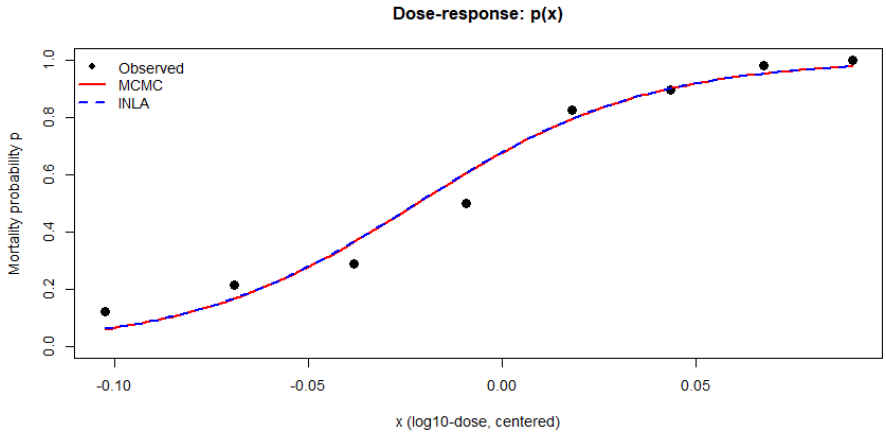


Figure 9: Mortality rate as a function of dose level. Observed data points are black dots, MCMC estimate in red and INLA in blue.

3 Problem 3

3.1 Task 3a)

The BYM2-model for spatial effect is a reparametrization of the BYM-model, which is the union of the besag model u^* and an iid model v^* [1]. The reparametrization is as follows,

$$x = \begin{pmatrix} \frac{1}{\sqrt{\tau}}(\sqrt{1-\phi}v + \sqrt{\phi}u) \\ u \end{pmatrix} \tag{11}$$

where both u and v have been standardized. The expression contains 2 hyper-parameters: τ and ϕ. τ is the marginal precision, while ϕ is the proportion of the marginal variance explained by u, also known as the mixing parameter [1]. τ controls the magnitude of the random effect $f(u_i)$ (high $\tau \rightarrow$ smaller random effects). ϕ determines how much of the variation is due to local noise compared to the spatial structure.

3.2 Task 3b)

In order to evaluate the covariate effect, we can look at the β value that the model found, which represents the relative risk per unit increase in x. The mean β (after exponentiating to reverse log-transform) value is, $\hat{\beta} = 1.0069$, while the 95%-CI is: $(1.0044, 1.0095)$. Note that since these values are greater than 1, we can conclude with 95% certainty that smoking proxy increases the larynx cancer mortality. The model summary is shown in figure 10.

In figure 11 we see the raw (left) and fitted (right) relative risks plotted on the map of Germany. One thing to note is that the RR on the raw map has a wider range, than the fitted map. Also, we note that the raw RR is very noisy as neighboring districts have vastly different RRs. The fitted map on the other hand is way smoother, because of the ICAR component of BYM2 which pulls extreme values towards the spatial mean. The southern part of Germany generally has a higher RR than the northern part and thus has a higher larynx cancer risk.

```
Time used:
    Pre = 17.4, Running = 0.993, Post = 0.121, Total = 18.5
Fixed effects:
             mean    sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) -0.383 0.066     -0.512   -0.383     -0.254 -0.383   0
x            0.007 0.001      0.004    0.007      0.009  0.007   0

Random effects:
  Name    Model
    region BYM2 model

Model hyperparameters:
                      mean    sd 0.025quant 0.5quant 0.975quant
Precision for region 34.80 7.955     22.010   33.848      53.16
Phi for region        0.88 0.099      0.617    0.909       0.99
                       mode
Precision for region 31.916
Phi for region        0.969

Deviance Information Criterion (DIC) ...............: 2743.25
Deviance Information Criterion (DIC, saturated) ....: 588.99
Effective number of parameters .....................: 98.62

Watanabe-Akaike information criterion (WAIC) ...: 2736.11
Effective number of parameters .................: 77.05

Marginal log-Likelihood:  -1109.43
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Figure 10: Model summary of BYM2 on Germany dataset.

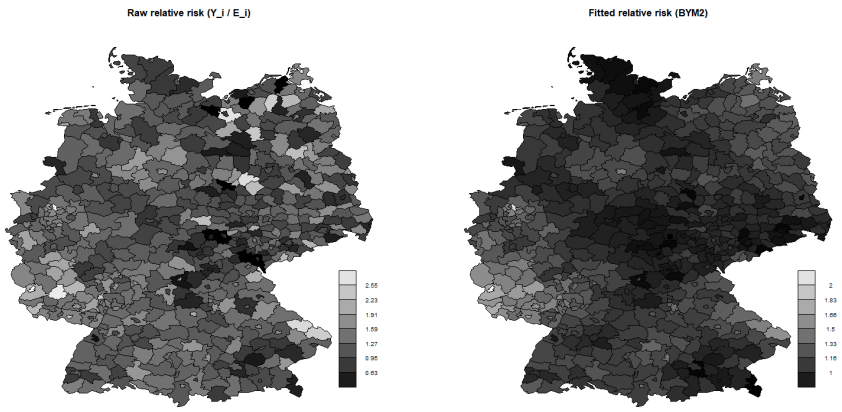


Figure 11: BYM maps of relative risk (Y_i/E_i). Left: Raw relative risk from dataset. Right: Fitted relative risk w/ BYM2.

3.3 Task 3c)

Section 2.4 in Gomez-Rubio mentions 5 criteria that are computed by INLA. First we have the marginal likelihood, which is the only one enabled by default. It tells us the probability of the observed data given a model $\mathcal{M}$: $\pi(\mathbf{y} \mid \mathcal{M})$. Next we have the Conditional Predictive Ordinates (CPO), which computes the probability of and observation y_i given all the rest of the data (without i). This is computed for each sample and summed up:

$$CPO = -\sum_{i=1}^{n} \log(CPO_i) = -\sum_{i=1}^{n} \log(\pi(y_i \mid y_{-i})). \tag{12}$$

Predictive Integral Transform (PIT) computes the probability of a new observation being less than or equal to the actual observed value (for every observation).

Lastly, we have the information based criterias, DIC and WAIC. Both of these take two things into account: how well the model fits the data and how complex the model is (number of parameters). What separates them is how the effective number of parameters is computed.

Higher values of marginal likelihood is desired, while we want lower values for DIC, WAIC and CPO. For PIT we ideally want a relatively flat/uniform histogram.

We try two different models, IID and Besag, which together make up the BYM model. In figure 12 we see the fitted RR maps using the two models. On the IID map we see that there isn't really any local structure on the map and neighboring districts have very varying values, which is what we expect since the IID

model doesn't incorporate the spatial structures. On the Besag map we see the opposite: large regions of similar RR values.

In terms of performance BYM2 had the best marginal likelihood (by a long-shot) and WAIC, while Besag had the best CPO and DIC (slightly better than BYM2 on both). IID performed significantly worse on all metrics. When taking all the criteria into account, BYM2 is overall the best model, which we would expect since it is a combination of the two others. From the PIT histograms in figure 13 there is not too much to read into. Visually, we see that none of the histograms are particularly close to being uniform.

Model	Marginal Likelihood	CPO	DIC	WAIC
BYM2	−1109.43	1376.91	2743.25	2736.11
IID	−1436.65	1421.80	2813.13	2814.50
Besag	−1609.97	1375.91	2740.91	2736.61

Table 3: Model assessment and choise criterias for BYM2, IID and Besag models.

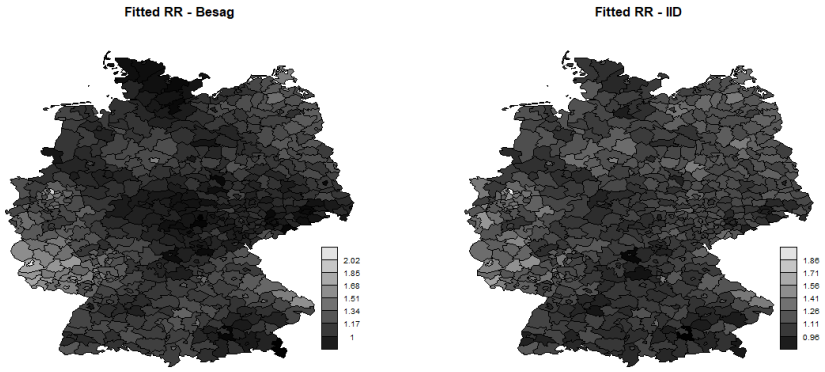


Figure 12: Fitted relative risk using Besag model (Left) and IID model (Right).

Code:

```
formula.besag <- Y ~ x + f(region, model = "besag", graph = g,
↪  scale.model = TRUE)
result.besag <- inla(formula.besag,
                     family = "poisson",
                     data = Germany, E = E,
                     control.compute = list(dic = T, waic = T, cpo
                     ↪  = T))
formula.iid <- Y ~ x + f(region, model = "iid")
```

```r
result.iid <- inla(formula.iid,family = "poisson",
                   data = Germany, E= E,
                   control.compute= list(dic = T, waic = T, cpo =
                   ↪  T))
lcpo.bym2  <- -sum(log(result\$cpo\$cpo), na.rm = TRUE)
lcpo.besag <- -sum(log(result.besag\$cpo\$cpo), na.rm = TRUE)
lcpo.iid   <- -sum(log(result.iid\$cpo\$cpo), na.rm = TRUE)
comparison <- data.frame(
  Model = c("BYM2", "Besag", "IID"),
  DIC   = c(result\$dic\$dic, result.besag\$dic\$dic,
  ↪  result.iid\$dic\$dic),
  WAIC  = c(result\$waic\$waic, result.besag\$waic\$waic,
  ↪  result.iid\$waic\$waic),
  LCPO  = c(lcpo.bym2,lcpo.besag, lcpo.iid),
  Mlik  = c(result\$mlik[1], result.besag\$mlik[1],
  ↪  result.iid\$mlik[1]))
```

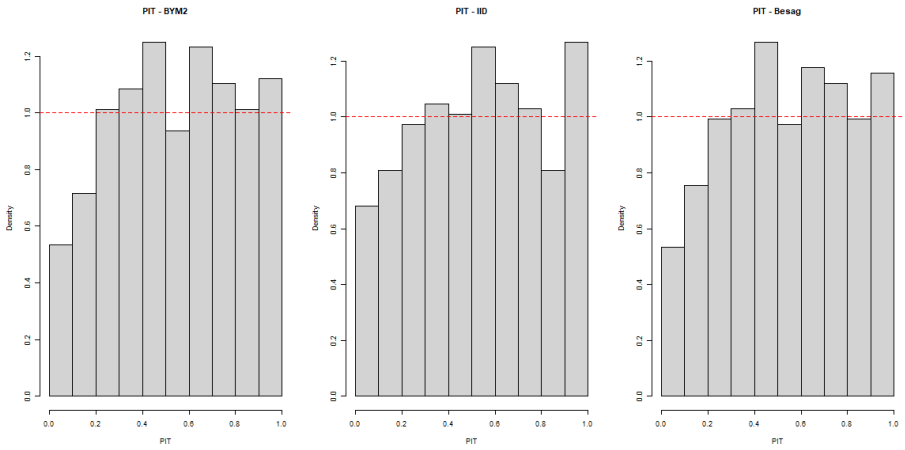


Figure 13: PIT histograms for the 3 tested models.

References

[1] Håvard Rue, Andrea Riebler, Sigrunn H. Sorbye, Janine B. Illian, Daniel P. Simpson, and Finn K. Lindgren. Bym2 model for spatial effects. Technical report, r-inla.org, 2016. Documentation of the BYM2 latent model in INLA.