Rousseau Mathieu, 67001800

UCLouvain
Belgium
??/??/????

1 Introduction

Cancer cell growth has been studied in vitro in 10 parallel experiments. We have a collection of data $\mathcal{D}_i$ where $i = 1, \dots, 10$ is the ith-experiment and where each $\mathcal{D}_i$ is a time series ranging from $t = 10$ to $t = 500$.

We describe the evolution of the tumor over time by a time-dependent Poisson distribution,

$$y(t) \sim \text{Pois}(\mu(t)) \tag{1}$$

where,

$$\mu(t) = \beta_0 \exp\left(\frac{\beta_1}{\beta_2}(1 - e^{-\beta_2 t})\right) \tag{2}$$

with $\beta_k > 0, \quad (k \in \{0, 1, 2\})$.

Notice we can reparametrize this model as,

$$\mu(t) = \alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t}) \tag{3}$$

with $\alpha_k > 0, \quad (k \in \{0, 1, 2\})$.

Questions

Question 1

a) Examine $\mu(t)$ and its relative change over time $\frac{1}{\mu(t)}\frac{\mathrm{d}\mu(t)}{\mathrm{d}t}$ in order to provide an interpretation for each parameter of (**??**).

Let's compute the derivative first,

$$\begin{aligned}\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} &= \mu(t)\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\beta_1}{\beta_2}(1 - e^{-\beta_2 t})\right) \\ &= \frac{\beta_1}{\beta_2}\mu(t)e^{-\beta_2 t}\beta_2 \\ &= \beta_1\mu(t)e^{-\beta_2 t}\end{aligned}$$

We then have,

$$\frac{1}{\mu(t)}\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} = \beta_1 e^{-\beta_2 t}$$

b) Provide the formulas connecting the β_k and α_k parameters.

Let's compute the relative change over time of **??**,

$$\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} = \mu(t)\alpha_1\alpha_2 e^{-\alpha_2 t}$$

We then have,

$$\frac{1}{\mu(t)}\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} = \alpha_1\alpha_2 e^{-\alpha_2 t}$$

Comparing with the two equations as well as the relative changes over time,

$$\alpha_1 \alpha_2 e^{-\alpha_2 t} = \beta_1 e^{-\beta_2 t}$$

$$\beta_0 \exp\left(\frac{\beta_1}{\beta_2}(1 - e^{-\beta_2 t})\right) = \alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t})$$

Question 2

a) Assuming independence between the data $\mathcal{D}_i$ collected during the experiment i where $\mathcal{D}_i = \{\, y_i(t_j) : j = 1, \ldots, J = 99 \,\}$ with $y_i(t_j) \sim \text{Pois}(\mu(t_j))$, provide an analytic form for the likelihood function $L(\vec{\alpha}|\mathcal{D}_i)$ for experiment i.

We know that the number of cancer cells $y_{i,j}$ for the experiment i at any given time t_j, $j \in \{\, 10, 15, 20, ..., 500 \,\}$ follow a Poisson distribution of parameter μ. Its probability mass function is given by,

$$p(y_{i,j}) \propto \mu^{y_{i,j}} \cdot e^{-\mu}$$

where an expression for μ is given above (**??**) as a function of $\alpha_k (k = 0, 1, 2)$.

Assuming independance, we then have,

$$\begin{aligned} L(\vec{\alpha}|\mathcal{D}_i) &= L((\alpha_0, \alpha_1, \alpha_2)|y_i) \\ &= \prod_{j=1}^{99} p(y_{i,j}) \\ &= \prod_{j=1}^{99} \left(\mu^{y_{i,j}} \cdot e^{-\mu}\right) \end{aligned}$$

b) Provide a R function enabling to compute the log-likelihood for given parameter values $\theta_k = \log(\alpha_k)$ and data D_i.

We have,

$$\begin{aligned} l(\vec{\alpha}|\mathcal{D}_i) &:= \ln(L((\alpha_0, \alpha_1, \alpha_2)|y_i)) \\ &\propto \sum_{j=1}^{99} \ln\left(\left[\mu^{y_{i,j}} \cdot e^{-\mu}\right]\right) \\ &= \sum_{j=1}^{99} \ln(\mu^{y_{i,j}} \cdot e^{-\mu}) \end{aligned}$$

```
log.mu <- function(alpha, t_j) {
  log(alpha[1]) - alpha[2] * exp(- alpha[3] * t_j)
}

log.likelihood <- function(theta, t, experiment) {
  n <- length(t)
  alpha <- exp(theta)

  result <- sum(sapply(1:n, function(j) {
    dpois(x = t[j], lambda = exp(log.mu(alpha, t[j])), log = TRUE)
  }))

  return(result)
}
```

c) Provide a R function enabling to compute the log-posterior $p(\vec{\theta}|\mathcal{D}_i)$ for given $\theta = \theta_0, \theta_1, \theta_2$, data $\mathcal{D}_i$ and large variance priors.

We have no information on the distribution of any parameter θ_k, only that it should have large variance. We will then use a non-informative prior. We could choose for example a Gamma distribution with paramaters $a = 1/2$ and $b = 0.001$ in order to have large variance priors but such a prior distribution received some critiscism in the litterature [1]. An alternative is to use a prior Normal distribution with a large variance of 100,

$$\theta_k \sim \mathcal{N}(\mu_0 = 0, \sigma_0^2 = 100) \tag{4}$$

Its PDF is given by,

$$f(\theta; \mu_0, \sigma_0) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\theta-\mu_0}{\sigma_0}\right)^2} \tag{5}$$

We find then,

$$\begin{aligned}
p(\vec{\theta}|\mathcal{D}_i) &\propto \ln(L(\vec{\theta}|\mathcal{D}_i) \cdot (P(\theta_0)P(\theta_1)P(\theta_2))) \\
&\propto l(\vec{\theta}|\mathcal{D}_i) + \ln\left(P(\theta_0)P(\theta_1)P(\theta_2)\right) \\
&\propto \sum_{j=1}^{99} \ln(\mu^{y_{i,j}} \cdot e^{-\mu}) + \ln(f(\theta_0; \mu_0, \sigma_0)) + \ln(f(\theta_1; \mu_0, \sigma_0)) + \ln(f(\theta_2; \mu_0, \sigma_0))
\end{aligned}$$

```
log.prior <- function(theta) {
  theta0 <- theta[1]
  theta1 <- theta[2]
  theta2 <- theta[3]

  prior.theta0 <- dnorm(theta0, mean = 0, sd = 10, log = TRUE)
  prior.theta1 <- dnorm(theta1, mean = 0, sd = 10, log = TRUE)
  prior.theta2 <- dnorm(theta2, mean = 0, sd = 10, log = TRUE)

  return(prior.theta0 + prior.theta1 + prior.theta2)
}

log.posterior <- function(theta, t, experiment) {
  log.likelihood(theta, t, experiment) + log.prior(theta)
}
```

d) Using the preceding R function, obtain the mean vector and variance-covariance matrix in the Laplace approximation to the marginal posterior $\boldsymbol{\theta}$ given the data from experiment 1.

1. http ://www.stat.columbia.edu/%7Egelman/research/published/taumain.pdf

```r
laplace.approximation <- function(log.posterior, inits, n_samples, ...) {
  fit <- optim(
    par = inits,
    fn = log.posterior,
    control = list(fnscale = -1),
    hessian = TRUE,
    ...
  )

  mean <- fit$par
  var_cov_matrix <- solve(-fit$hessian)
  samples <- rmvnorm(20000, mean, var_cov_matrix)

  return(list(
    mean = mean,
    var_cov_matrix = var_cov_matrix,
    samples = samples
  ))
}

inits <- c(theta0 = 0.001, theta1 = 0.001, theta2 = 0.001)
lapprox <- laplace.approximation(log.posterior, inits, 10000, t = df$day, experiment = df$Exp1)
```

We get the following mean vector $\bar{\boldsymbol{\theta}}$ and variance-covariance matrix $\Sigma_{\boldsymbol{\theta}}$,

$$\bar{\boldsymbol{\theta}} = \begin{pmatrix} 6.380560 & 1.168446 & -5.224318 \end{pmatrix}^T \tag{6}$$

$$\Sigma_{\boldsymbol{\theta}} = \begin{pmatrix} 1{,}002\,258 \times 10^{-3} & -9{,}989\,023 \times 10^{-5} & -0{,}001\,152\,835\,2 \\ -9{,}989\,023 \times 10^{-5} & 2{,}030\,509 \times 10^{-4} & 0{,}000\,262\,259\,4 \\ -1{,}152\,835 \times 10^{-3} & 2{,}622\,594 \times 10^{-4} & 0{,}001\,486\,148\,8 \end{pmatrix} \tag{7}$$

Question 3

a) Using a **Metropolis algorithm** (*with componentwise proposals*) and the R software (without using JAGS or other specific packages), draw a random sample from the posterior distribution of $(\vec{\theta}|\mathcal{D}_1)$.

```r
metropolis.cw <- function(theta.init, sd.proposals, t, experiment, n_steps) {
  pb <- progress_bar$new(
    format = "metropolis algorithm [:bar] :percent eta: :eta",
    total = n_steps,
    clear = FALSE,
    width = 60
  )

  n_parameters <- length(theta.init)

  # matrix of dim (n_steps, n_parameters)
  theta <- matrix(nrow = n_steps, ncol = n_parameters)
```

```r
colnames(theta) <- c("theta0", "theta1", "theta2")

# initialize
theta[1, ] <- theta.init

# counter
n_accept <- rep(0, n_parameters)

pb$tick(0)

for (n in 2:n_steps) {
  pb$tick()

  # get all the current parameters for the step n
  theta.current <- theta[n - 1, ]

  for (i in 1:n_parameters) {
    # sample a proposal for parameter i from a normal distribution centered at the
    ↪  previous parameter point with a given standard deviation
    theta.proposed_i <- theta.current[i] + rnorm(1, mean = 0, sd =
    ↪  sd.proposals[i])

    # update the current parameters vector with the proposal
    theta.proposed <- theta.current
    theta.proposed[i] <- theta.proposed_i

    proba <- min(
      1,
      exp(log.posterior(theta.proposed, t, experiment) -
      ↪  log.posterior(theta.current, t, experiment))
    )

    accept <- sample(0:1, 1, prob = c(1 - proba, proba))

    if (accept) {
      n_accept[i] <- n_accept[i] + 1
      # update the final vector
      theta[n, i] <- theta.proposed_i

      # update the temporary current vector for the next iteration
      theta.current[i] <- theta.proposed_i
    } else {
      # update the final vector
      theta[n, i] <- theta.current[i]

      # note: the value of the temporary current vector for this parameter
      # remains the same in that case
    }
  }
}

# note: n_steps - 1 because there is theta.init "automatically accepted"
# (we do n_steps - 1) iterations
```

~~~
  accept.rate <- round(n_accept / ((n_steps - 1)), 2)
  return(list(theta = theta, accept.rate = accept.rate))
}
```

```{r}
#sd.proposals <- c(0.017, 0.28, 0.023)
sd.proposals <- c(0.017, 0.025, 0.023)

metropolis.results <- metropolis.cw(
  theta.init = c(5.5, 0.98, 0.01),
  sd.proposals = sd.proposals,
  t = df$day,
  experiment = df$Exp1,
  n_steps = N_STEPS
)

# goal is to reach an acceptance rate of 0.4
metropolis.results$accept.rate
~~~

For the single component metropolis algorithm, we should aim for $\approx 40\,\%$ of acceptance rate. We ran the algorithm for 140 000 steps and a burnin of 5000 steps. In order to have such an acceptance rate for each θ parameter, we tweaked the sigma proposals to finally start with $\sigma = (0.017, 0.021, 0.019)$ giving us an acceptance rate of $(40\,\%, 40\,\%, 40\,\%)$ respectively for $(\theta_0, \theta_1, \theta_2)$.

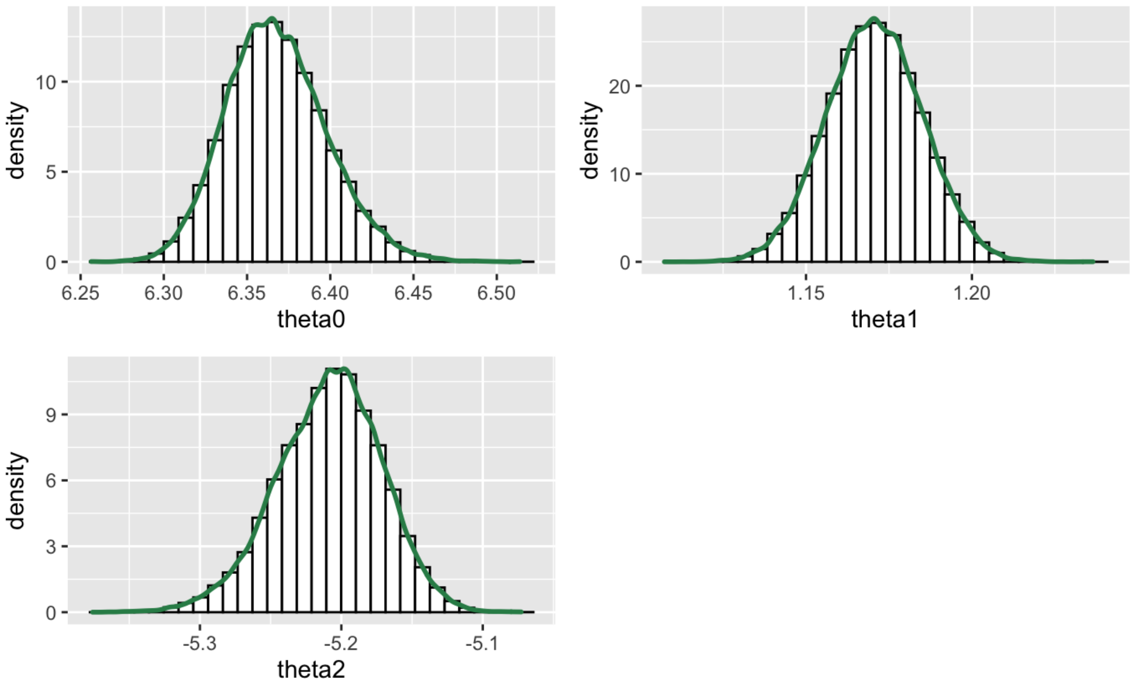

Figure 1 – *Random sample from the posterior distribution of $(\vec{\theta}|\mathcal{D}_1)$ using a component-wise metropolis algorithm*

b) Evaluate the convergence of your algorithm. After a suitable burnin, evaluate the effective sample size for each θ_k and make sure that each of these values is at least 500.

Since there does not exist any statistical test that would be a proof of convergence, we will perform multiple tests or analysis in order to increase our confidence in the fact that our algorithm has effectively converged.

Visual analysis

We can first perform a visual analysis to check for the convergence of the metropolis algorithm.

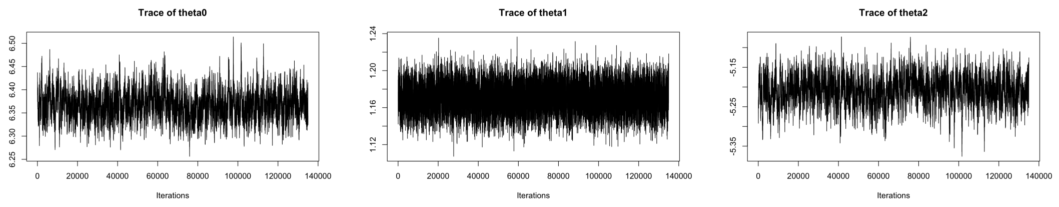

FIGURE 2 – *Traceplots of the θ_k parameters*

The mixing is good for all the parameters and in particular for θ_1. However, we can go further and use some statistical tools to increase our confidence in the convergence of these samples.

Gelman-Rubin statistic

Imagine we would start multiple chains in parallel with different starting values. Then they all should converge to the stationnary distribution so that after some amount of time, one could not distinguish between the different chains. We can assess such a thing by comparing the variation between chains. If all chains converge toward the stationnary distribution, they all should be the same and therefore the variation between chains should be zero. This can be assessed with the Gelman-Rubin statistic.

Let $\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_l}$ be different samples from Markov chains run in parallel with different initial values. As the length of the chains approaches infinity and the between chains variance approaches zero, the Gelman-Rubin statistic ($\hat{R}$) approaches the value of 1. Practically, we can aim for a value of $\hat{R}$ in the neighborhood of 1.1.

Here, we created three chains using different initial parameters for $\boldsymbol{\theta}$. The $\hat{R}$ of each parameter is almost equal to 1.0 with an upper confidence interval below 1.1.

parameters	$\hat{R}$	upper C.I.
θ_0	1.001888	1.006282
θ_1	1.000398	1.001150
θ_2	1.002163	1.006658

TABLE 1 – *Results of the Gelbman-Rubin test*

Moreover, we can notice below that the convergence appears after roughly 5000 iterations for each parameter.

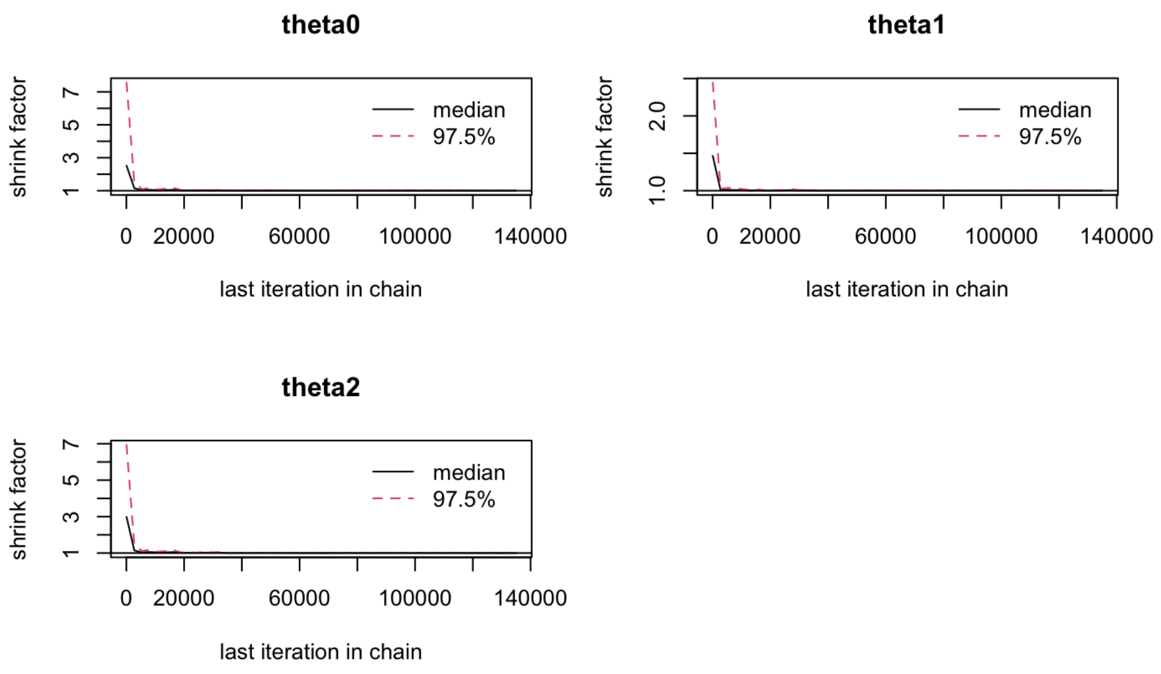

Figure 3 – *Evolution of the Gelman-Rubin shrink factor with the number of iterations of the metropolis algorithm (component-wise)*

Geweke statistic

The Geweke statistic has the particularity to assess convergence from a single chain. This statistic will perform a two mean test comparing the mean of the first 10 % of the chain with the last 50 %. If convergence has occured, the two means should be equal. Practically, we use a classical z-score test from frequentist statistic based on the effective sample sizes.

parameters	z-score	p-value
θ_0	-1.3259	0.9075636
θ_1	-0.5802	0.7191101
θ_2	1.1019	0.1352526

Table 2 – *Results for the Geweke statistic (z-score)*

We notice that at a 5 % significance level, one cannot reject the hypothesis that the means of the two subchains are equal the z-score. To conclude, none of these statistcs showed us that our algorithm did not converge into the stationnary distribution. We can be confident and go further.

Effective sample sizes

Computing the effective sample size (*i.e. the number of samples that would be generated if there were no autocorrelation*) for each parameter after a burnin of 5000 iterations, we can confirm that we have at least 500 samples for each θ_k.

θ_0	θ_1	θ_2
683.7875	4121.8599	648.4315

TABLE 3 – *Effective sample sizes*

c) Compare the observed count data series for experiment 1 with the fitted curve for $\mu(t)$

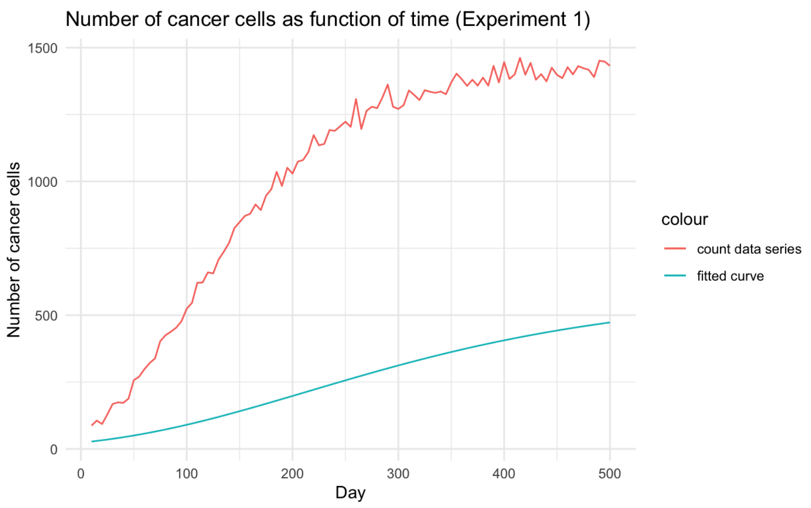

FIGURE 4 – *Comparison of the observed count data series for experiment 1 (black) with the fitted curve for $\mu(t)$ (red)*

On the plot above, it seems like the fitted curve for $\mu(t)$ computed with the mean parameters of $\boldsymbol{\theta}$ — that were obtained with the component-wise metropolis algorithm ran for the first experiment — underestimate a lot the growth of the number of cancer cells as function of time.

d) Provide point estimates and 95 % credible regions for β_0, β_1, β_2.

From the obtained sample from the metropolis algorithm, we can compute the mean and median of the $\boldsymbol{\theta}$ parameters as well as their respective 95 % credible regions. Using the relations found in the first question, we can then find these quantity for the $\boldsymbol{\beta}$ parameters. The results are summarized in the tables below.

Parameter	Median	Mean
β_0	6.37	6.37
β_1	1.17	1.17
β_2	-5.21	-5.21

TABLE 4 – *Point estimates of $\boldsymbol{\beta}$*

Parameter	Lower	Upper
β_0	6.31	6.42
β_1	1.14	1.19
β_2	-5.28	-5.13

TABLE 5 – 95 % *credible region for $\boldsymbol{\beta}$*

e) Provide a point estimates and 95 % credible region for the expected number of cancer cells at time $t = 0$ and $t = +\infty$ for experiment 1.

Question 4

a) Answer the same question as in Q3 using a Metropolis algorithm with vector proposals for $\boldsymbol{\theta}$ and making use of the variance-covariance matrix from the Laplace approximation in Q2d. Use the same number of iterations (after burnin) as in Q3.

In order to get 20 % of acceptance rate, we multiplied the variance obtained from the variance-covariance matrix by a factor of 20.

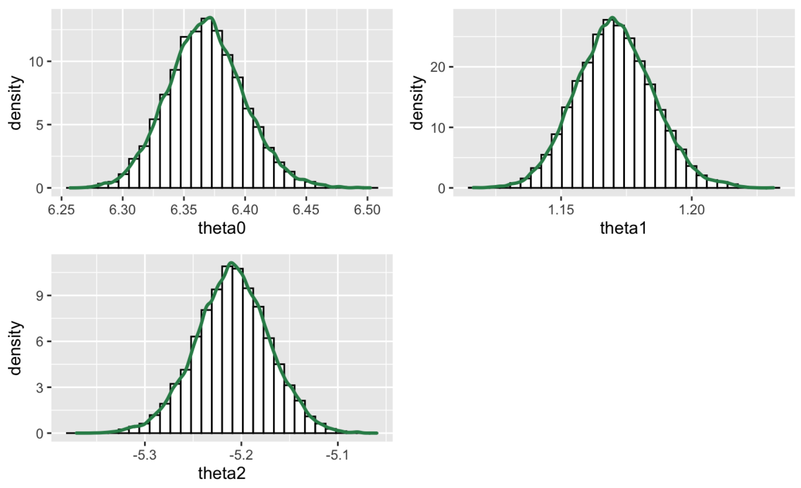

Figure 5 – *Random sample from the posterior distribution of $(\vec{\theta}|\mathcal{D}_1)$ using a vector-wise metropolis algorithm*

Convergence

From the visual analysis, we have a good mixing which is a good signal of convergence.

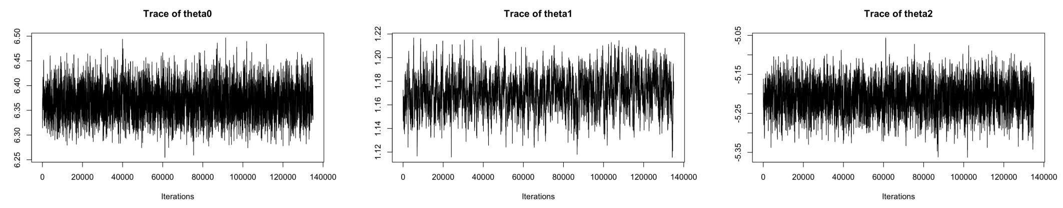

Figure 6 – *Traceplots of the θ_k parameters*

Moreover, the $\hat{R}$ computed from the Gelman statistic is below 1.1 and we notice from the Geweke test that at a 5 % significance level, we cannot reject the null hypothesis that the mean of the two computed subchains are equal for each parameter. Therefore we have a strong confidence in the convergence of the metropolis vector-wise algorithm.

parameters	$\hat{R}$	upper C.I.
θ_0	1.000513	1.000901
θ_1	1.001974	1.002180
θ_2	1.000653	1.001020

Table 6 – *Results of the Gelbman-Rubin test*

parameters	z-score	p-value
θ_0	-0.4956	0.6899117
θ_1	-1.7949	0.9636652
θ_2	-0.4138	0.6604897

Table 7 – *Results for the Geweke statistic (z-score)*

Finally, we have at least 500 effective samples for each parameter.

θ_0	θ_1	θ_2
1682.979	504.130	1282.254

Table 8 – *Effective sample sizes*

b) Compare your results with the previous ones.

Fitted curve

The mean for each $\boldsymbol{\theta}$ parameter is the same as for the component-wise algorithm. Therefore, as expected, the growth rate of cancer cells is again underestimate by our model.

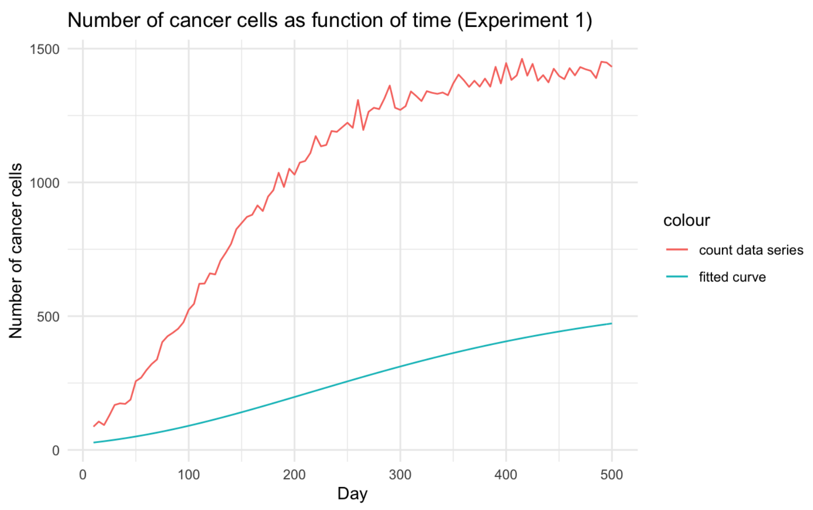

Figure 7 – *Comparison of the observed count data series for experiment 1 (black) with the fitted curve for $\mu(t)$ (red)*

Point estimate and $95\,\%$ credible region

Following what we said above, obviously we get the point estimate and credible regions for the $\boldsymbol{\beta}$ parameters.

Parameter	Median	Mean
θ_0	6.37	6.37
θ_1	1.17	1.17
θ_2	-5.21	-5.21

Table 9 – *Point estimates of $\boldsymbol{\theta}$ parameters*

Parameter	Lower	Upper
θ_0	6.31	6.42
θ_1	1.14	1.19
θ_2	-5.28	-5.13

Table 10 – $95\,\%$ *credible region for $\boldsymbol{\theta}$ parameters*

Question 5

Sample $p(\boldsymbol{\theta}|\mathcal{D}_1)$ using **JAGS** with the same number of iterations (after burnin) as in Q3.

a) Compare your results with the previous ones.

Fitting a JAGS model, we now correctly estimate the growth rate of the number of cancer cells over time as we can notice on the plot below.

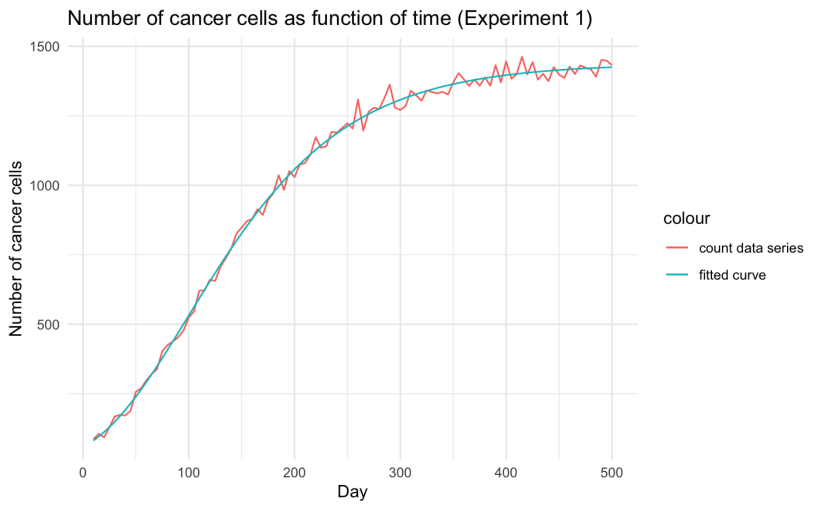

FIGURE 8 – *Comparison of the observed count data series for experiment 1 (black) with the fitted curve for* $\mu(t)$ *(red)*

b) Sample the predictive distribution of the number of cells at time $t = +\infty$ and report a point estimates and a credible interval for that quantity.

Question 6

Consider now the data from the 10 experiments.

a) Fit, using **JAGS**, a hierarchical model with $\theta_k = \log \alpha_k$, $(k = 0, 1, 2)$ randomly varying between experiments. Report your results in a meaningful manner.

b) Compare your estimates (+ *credible regions*) for the θ_k parameters associated to experiment 1 in this hierarchical model with the ones obtained in Q5.

c) Sample the predictive distribution of the number of cells at time $t = +\infty$ for experiment 1 using the hierarchical model and report a point estimates and a credible interval for that quantity. Compare your results with answers in Q5b.

Appendix