Modeling Cancer Cell Growth: Bayesian Analysis and Inference

LSTAT2130 - Introduction to Bayesian statistics

Victor Dujardin, Lucas Elvira and Mathieu Graf

Professor: Philippe Lambert
Teaching Assistant: Hortense Doms

May 2023

Contents

Introduction **2**

Question 1 **4**

1 a) 4

1 b) 4

Question 2 **5**

2 a) 5

2 b) 5

2 c) 6

2 d) 6

Question 3 **8**

3 a) 8

3 b) 9

3 c) 11

3 d) 12

3 e) 12

Question 4 **12**

4 a) 12

4 b) 18

Question 5 **19**

5 a) 19

5 b) 22

Question 6 **22**

6 a) 22

6 b) 23

6 c) 23

Conclusion **23**

Appendix **24**

Introduction

This project involves the study of cancer cell growth using Bayesian statistics. We are given data from ten separate experiments, each providing a time series of observed cancer cell counts. The main objective of our analysis is to model the evolution of these cell counts over time, with the understanding that the counts are not available at time 0.

The cancer cell growth is modeled using a Poisson process with a time-varying rate parameter, $\mu(t)$. This rate parameter is described by an exponential growth function, and it can be reparametrized in two different ways, both involving three parameters. We will investigate these parameters and their relative changes over time to interpret their meanings. Moreover, we will derive the formulas connecting the parameters from the two reparametrizations.

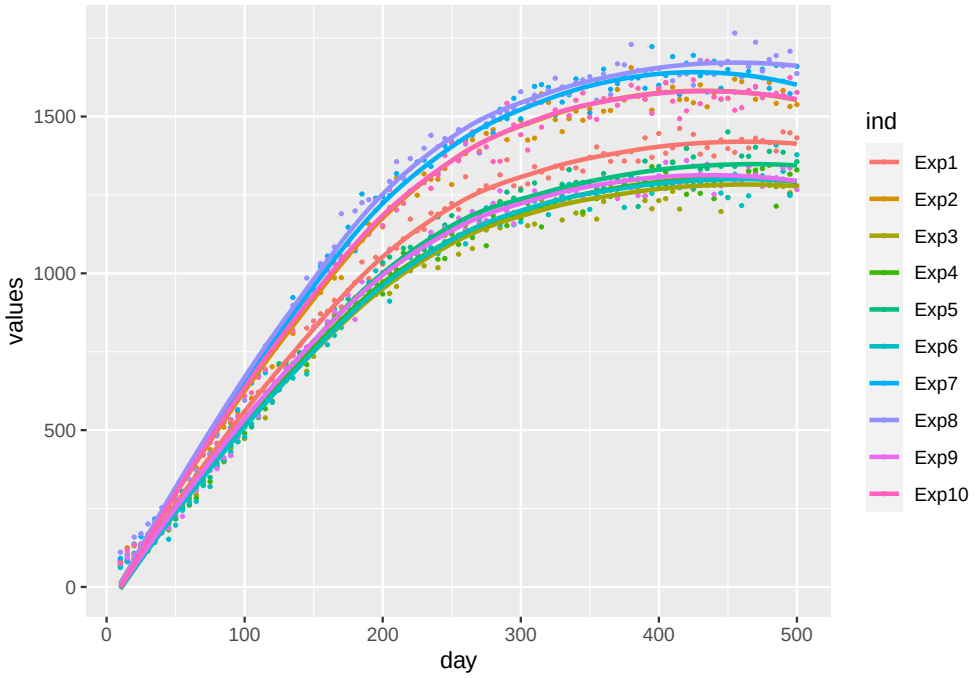
To analyze the data, we will assume independence between the observations from different experiments. We will then derive an analytic form for the likelihood function for a single experiment. With this likelihood function, we will use the R programming language to compute the log-likelihood and log-posterior functions, which will guide our Bayesian inference. These functions will be central to our subsequent analysis.

We will use the Metropolis algorithm, a Markov Chain Monte Carlo (MCMC) method, to draw samples from the posterior distribution of the parameters. Using this sample, we will evaluate the convergence of our algorithm and provide point estimates and credible regions for the parameters. Additionally, we will compare the observed count data with the fitted curve of $\mu(t)$ to evaluate the fit of our model.

Finally, we will expand our analysis to consider the data from all ten experiments. We will fit a hierarchical model using the JAGS software, allowing the parameters to vary randomly between experiments. This hierarchical model will provide a more nuanced understanding of the cell growth process, taking into account the variations between different experimental runs.

Throughout this project, we will be using the Bayesian approach to statistical inference, where uncertainties about the unknown parameters are characterized by probability distributions. This allows us to quantify our uncertainty about the parameters and make probabilistic statements about their values based on the observed data. We will also emphasize reproducibility in our analysis, ensuring that all our results can be reproduced using the same random seed.

To first see the data we are about to analyse, here is a plot of the different measures of each experiment over time:



We observe similarities in the shape and behavior of our data, but slight differences emerge during the growth phase. To investigate and define the coefficients of our model, we analyze each experiment's results. The model is expressed by μ as the following :

$\mu(t)$ can be expressed as these two functions (Appendix: Code I.1)

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

$$Or$$

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

Question 1

1 a)

The relative change over time $\frac{1}{\mu(t)}\frac{d\mu(t)}{dt}$ can be computed in the following way:

Firstly, given the function $\mu(t)$:

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

we can compute its derivative $\frac{d\mu(t)}{dt}$ by applying the chain rule:

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

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

Hence, the relative change over time is:

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

At $t = 0$, we have $\mu(t) = \beta_0$ and the relative change is $-\beta_1$.

As $t \to \infty$, $\mu(t)$ approaches $\beta_0 e^{\beta_1/\beta_2}$ and the relative change tends towards 0. This is consistent with the intuition that the rate of increase in the data diminishes as time progresses.

In this model, the parameters β_0, β_1, and β_2 have the following interpretations:

β_0 represents the mean cell count at $t = 0$. This is essentially the starting point of the system, or the "baseline" cell count. As a scaling factor in the expression for $\mu(t)$, a larger β_0 means a higher predicted cell count at any given future time. However, β_0 does not affect the relative rate of change over time, as this is a relative measure.

β_1 can be thought of as the initial growth rate, or the instantaneous relative rate of change at $t = 0$. This parameter also influences the equilibrium state (as $t \to \infty$), since the mean number of cells at equilibrium is influenced by the initial growth power.

β_2 represents how quickly the relative growth rate decreases over time. Essentially, it is a damping factor. A larger β_2 implies a faster decrease in the relative growth rate over time, leading to fewer cells at equilibrium.

Finally, the ratio β_1/β_2 is a descriptor of the cells' overall growth rate. A larger ratio indicates faster cell growth.

1 b)

We start by equating two models:

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

With using $t = 0$ and $t = +\infty$, we replace and with some calculation we found the following equations :

$$\alpha_0 = \beta_0 e^{\frac{\beta_1}{\beta_2}}$$
$$\alpha_1 = \frac{\beta_1}{\beta_2}$$
$$\alpha_2 = \beta_2$$

Question 2

2 a)

The likelihood can but computed as

$$L(\theta) = \prod_{i=1}^{n} f_i(y_i|\theta)$$

We know that $y_i(t_j) \sim \text{Pois}(\mu(t_j))$, which means that $P(Y(t) = y|\mu(t)) = \frac{e^{-\mu(t)}\mu(t)^y}{y!}$

$$L(\mu(t)|D_i) = \prod_{j=1}^{99} \left[\frac{e^{-\mu(t_j)}\mu(t_j)^{y_{ij}}}{y_{ij}!} \right]$$

$$L(\alpha|D_i) = \prod_{j=1}^{99} \left[\frac{e^{-\alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t_j})}[\alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t_j})]^{y_{ij}}}{y_{ij}!} \right]$$

2 b)

The Log-Likelihood is:

$$LL(\alpha|D_i) = \sum_{j=1}^{99} \left[-\alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t_j}) + y_{ij} \log(\alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t_j})) - \log(y_{ij}!) \right]$$

$$= \sum_{j=1}^{99} \left[-\alpha_0 \exp(-\alpha_1 e^{-\alpha_2 t_j}) + y_{ij}(\log(\alpha_0) - \alpha_1 e^{-\alpha_2 t_j}) - \log(y_{ij}!) \right]$$

If $\theta_k = \log \alpha_k \Leftrightarrow e^{\theta_k} = \alpha_k$,

$$\Rightarrow LL(\theta|D_i) = \sum_{j=1}^{99} \Big[-\exp(\theta_0) \exp(-\exp(\theta_1) e^{-\exp(\theta_2) t_j}) + y_{ij}(\theta_0 - \exp(\theta_1) e^{-\exp(\theta_2) t_j}) - \log(y_{ij}!) \Big]$$

The log-likelihood function is a sum of logs of poisson distributions of parameter $\mu = e^{\theta_0} exp(-e^{\theta_1} e^{-e^{\theta_2} t})$

The R function is given in the appendix (Appendix: Code Q2b)

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

  mu = exp(theta0)*exp(-exp(theta1)*exp(-exp(theta2)*t))
  ll <- sum(dpois(Di,lambda = mu, log=TRUE))

  return(ll)

}
```

2 c)

By choosing a prior as a multivariate uniform distribution with a large variance, we can compute the log-posterior function as the following (Appendix: Code Q2c) :

$$\pi(\theta|D_i) \propto L(\theta|D_i) * \pi(\theta) \leftrightarrow log(\pi(\theta|D_i)) = C^{st} + LL(\theta|D_i) + log(\pi(\theta)) Where \theta \sim U(-\infty, \infty)$$

```
logprior = function(theta){
  theta0 = theta[1]
  theta1 = theta[2]
  theta2 = theta[3]
  log(dunif(theta0, min = -1e6, max = 1e6) *
        dunif(theta1, min = -1e6, max = 1e6) *
        dunif(theta2, min = -1e6, max = 1e6))
}


logpost = function(theta){
   loglike(theta) + logprior(theta)
}
```

2 d)

The laplace approximation for a posterior consists of fitting a normal distribution to a posterior where the mean is the maximum of the posterior distribution and the variance is the inverse observed Fisher Information.

$$(\theta|D) \sim N(\mu, \Sigma)$$

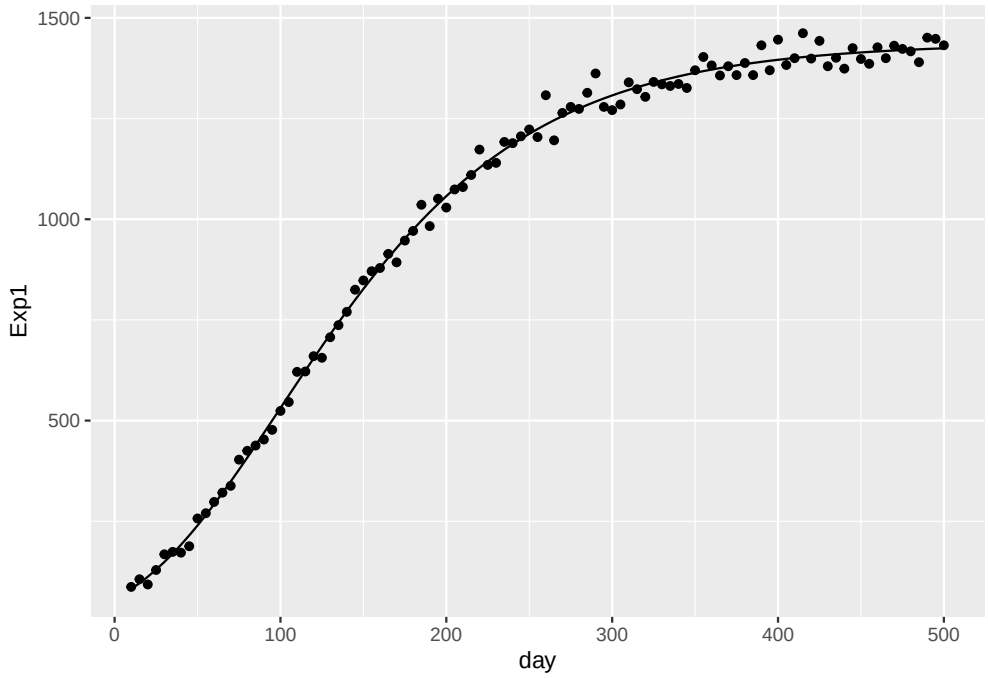
where $\mu = \hat{\theta}$ and $\Sigma = \mathbf{J}(\hat{\theta})^{-1}$

In order to get the mean vector in the Laplace approximation to the marginal posterior of θ we need find the mode of the logposterior function. We can do that by performing an optimization (Appendix: Code Q2d).

The mean vector is

```
[1]  7.270567  1.171298 -4.442964
```

We can see that our model fits the data.



The variance-covariance matrix is the inverse of the hessian of the log-posterior function which is already computed by R

```
              Theta0        Theta1        Theta2
Theta0  2.638132e-05 -2.111294e-05 -5.217265e-05
Theta1 -2.111294e-05  1.761176e-04  1.491314e-04
Theta2 -5.217265e-05  1.491314e-04  2.102476e-04
```

Question 3

3 a)

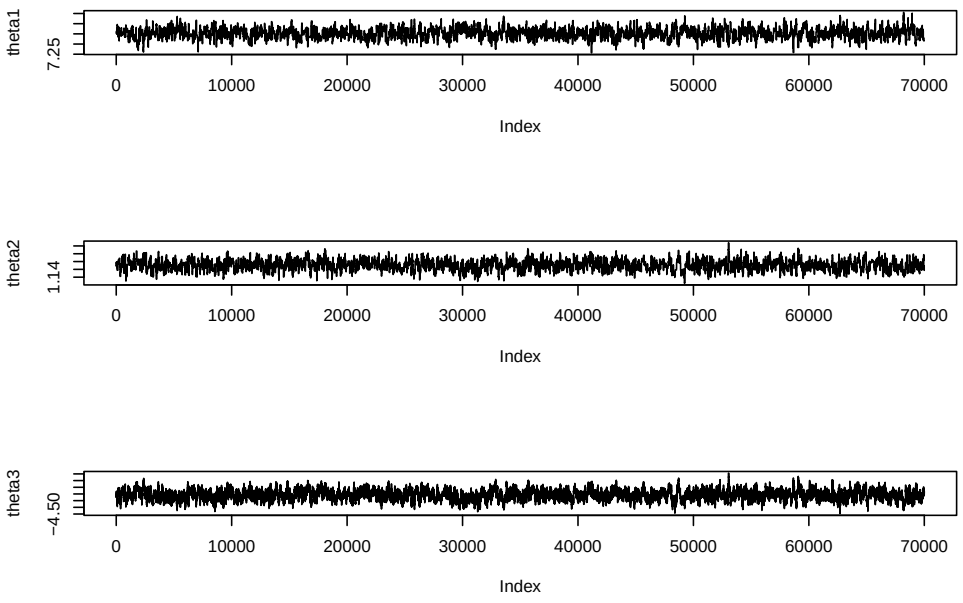
We now want to have a random samples for the three θ's to make inference. Firstly, we will get these θ's one at a time. To do so, we will use a Metropolis algorithm by using a Normal as symmetric proposal distribution and the MLE of θ as starting values. We will update the value of the θ's at each iteration, beginning with θ_0, and calculate a probability to know if we're doing better with the new θ or not. If it's the case, we give the new proposal value to θ and continue with the next parameter, else we stay with the actual value. To take a look at the performance of our algorithm, we count the number of times that a new proposal is accepted and compute the *acceptance rate*, which is the ratio between this number and the number of iterations that we've done. For a componentwise metropolis, we want an acceptance rate of 0.4 and for a vector proposal metropolis (see Section 4), we want 0.2. To reach this value, we will change the standard deviation of the symmetric proposal (here a Normal distribution) to meet our requirements.

We will use 70000 iterations to get a good sample of our θ's and use different standard deviation for each θ (Here we've used 0.07, 0.13 and 0.026 for θ_0, θ_1 and θ_2 respectively) (Appendix: Code Q3a).

And we get an acceptance rate of

```
[1] 0.3962485
```

Here are the traceplots of the different θ's :

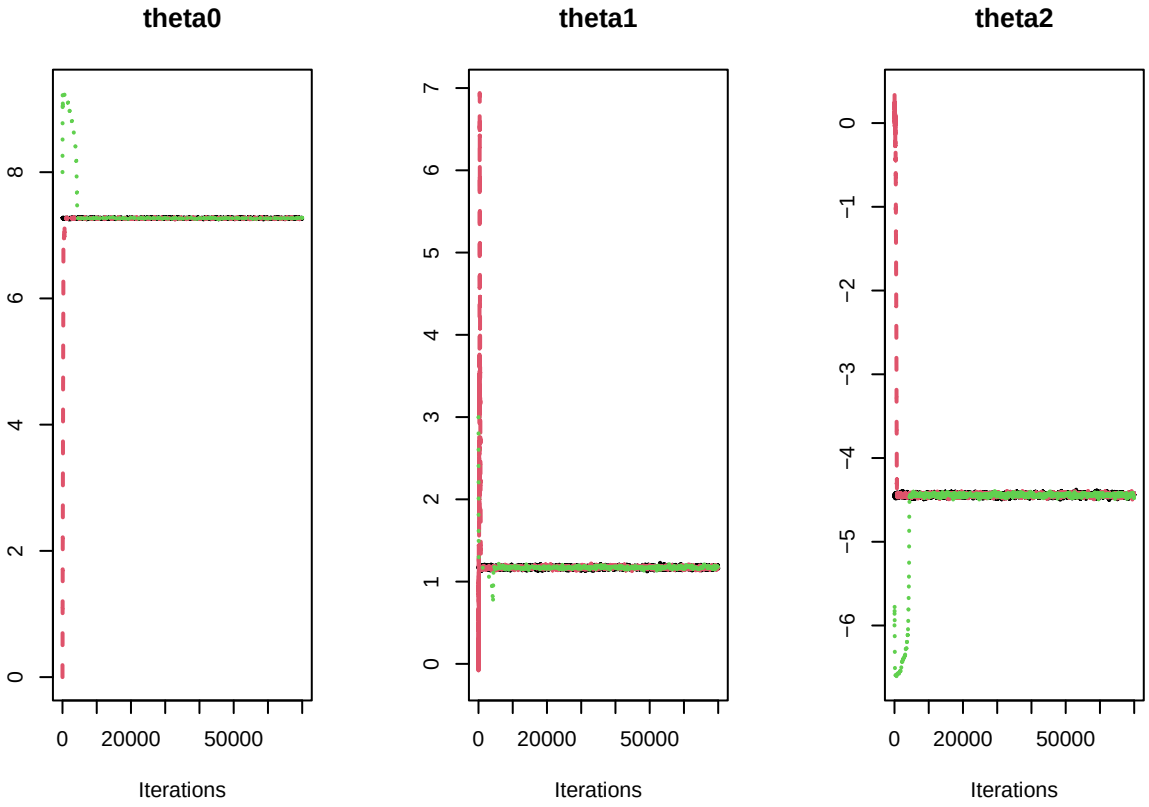


Here we obtain an acceptance rate of 0.396 so the target is achieved but the standard deviations needed are really low. It means that the posterior distribution (so the likelihood) change drastically with the value of theta, and so the real value of theta is easily detectable from other values.

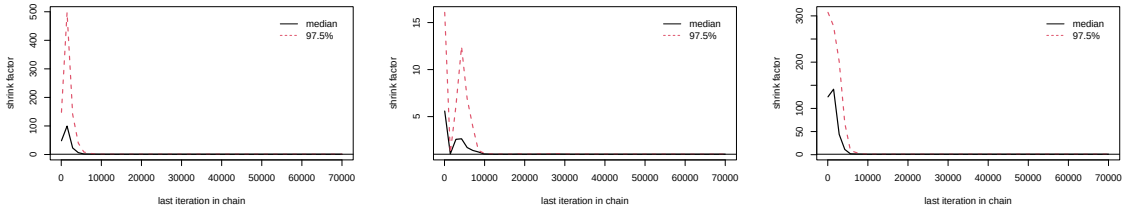
3 b)

To evaluate the convergence of our chain, we will use two techniques commonly used, the Gelman-Rubin diagnostic and the Geweke diagnostic. The first need to evaluate the algorithm at several different starting values for the parameters in order to evaluate if these different chains are equivalent (here we test 3 different values). It will be used to estimate at which point the convergence may have occur. The second technique splits the chain in 2 blocks and tests the equivalence of the mean between a part of the first block and the second block. If the test is significant, we can hope that occurence have occured. We used the term "hope" because these techniques can't prove convergence. We've used the same number of iterations than before and the same standard deviations. We've used two different vectors for the θ's : (0,0,0) and (8,3,-6) with respect to the signs of the MLE (Appendix: Code Q3b-1).

Here are the plots of the different chains for each θ:



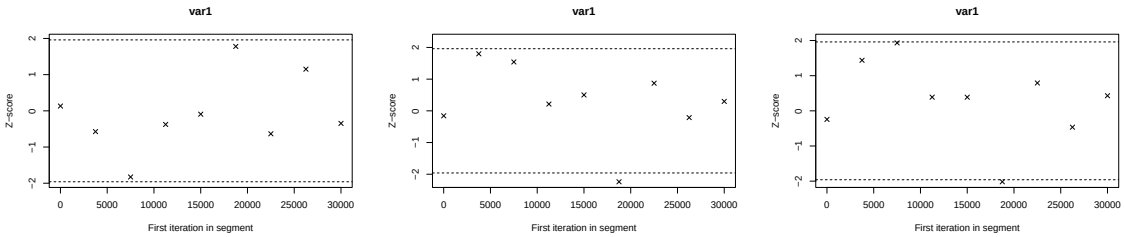
And the corresponding Gelman plots:



For the 3 thetas the convergence seems to occur after 10000 iterations. This convergence occurs faster for the first chain than the second. For the rest of the analysis, we will discard the 10000 first iterations to make sure that the sample that we have represents greatly the true value of the θ's.

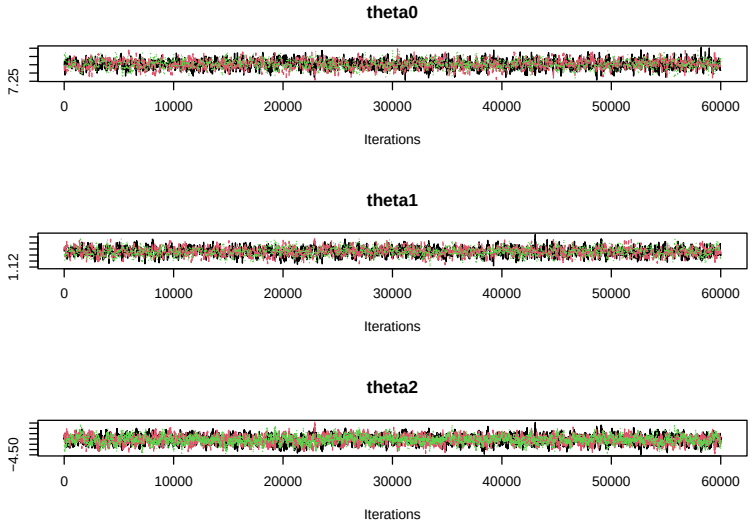
We're doing now the Geweke diagnostic (Appendix: Code Q3b-2). Here are the geweke plots and the effective sizes of our sample (θ_0, θ_1 and θ_2 respectively):

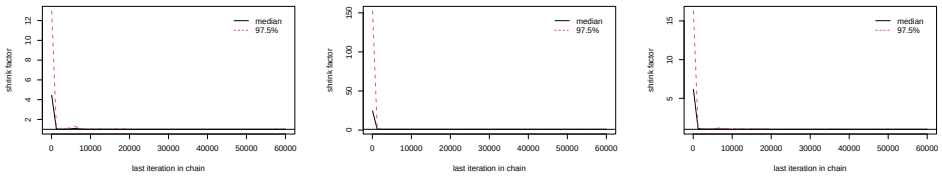
```
    var1
769.1923

    var1
674.4013

    var1
547.1961
```



The geweke diagnostic seems to show that our sample stay similar with the iterations and we don't reject the null hypothesis a large number of times (don't forget that there is always 5% of chance to reject the null hypothesis even if it's true). With an effective sample size above 500, we are sure to have an effective sample.

Here are the traceplots, Gelman plots and Gelman $\hat{R}$ of the thinned samples:



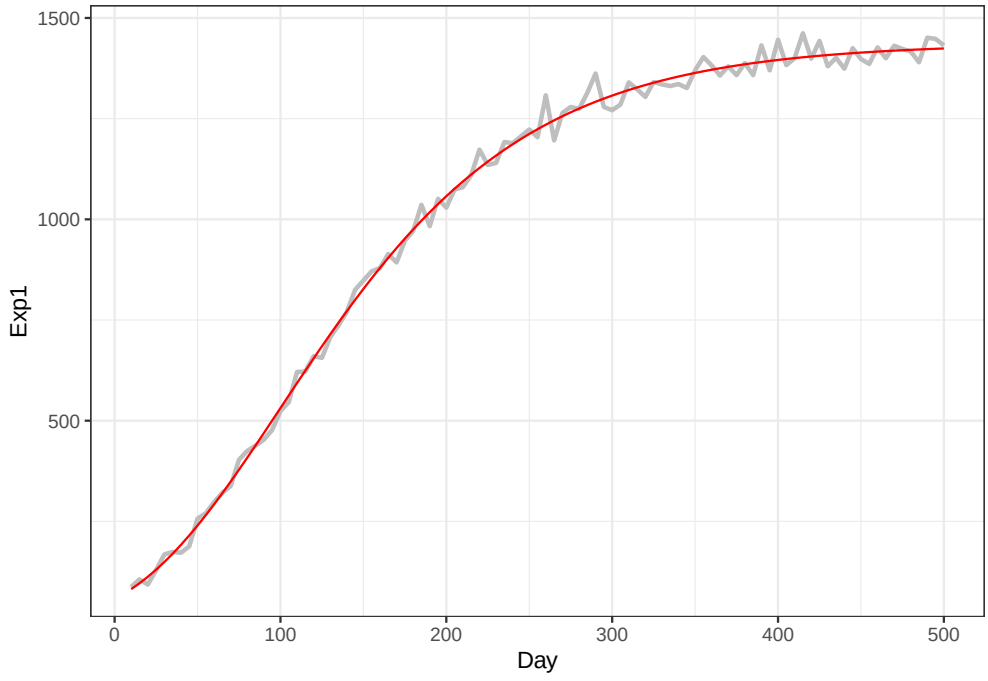



```
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1       1.02

Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1          1

Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1          1
```

The gelman statistics are all really close to 1, indicating that convergence may have occur.

3 c)

Now we will compare the observed count data series for Experiment 1 with the fitted curve for $\mu(t)$ by using the sample we've created. Here is a plot comparing the data we have and the curve computed with our sample (Appendix: Code Q3c):



Seems that our sample of θ's represents quite well the true value of the θ's.

3 d)

Now we will provide point estimates and 95% credible regions for β_0, β_1, β_2. To do so, we will use the sample of theta to compute the alpha's and then the beta's by the formulas calculated earlier in section 1 (Appendix: Code Q3d).

	2.5%	50%	97.5%
Beta0	52.1191389	57.1366311	62.3908190
Beta1	0.0360160	0.0379389	0.0399494
Beta2	0.0114241	0.0117650	0.0120985

The distribution of β_1 and β_2 are really centered around their mean.

3 e)

Idem but for the expected number of cancer cells at time $t = 0$ and $t = +\infty$ for Experiment 1 (Appendix: Code Q3e).

	x
2.5%	52.11914
50%	57.13663
97.5%	62.39082

We can see that the interval is equal to the β_0 interval because at $t = 0$, $\mu(t)$ become equal to β_0.

	x
2.5%	1423.099
50%	1436.855
97.5%	1452.741

Again, the interval is equal to those of α_0.

Question 4

4 a)

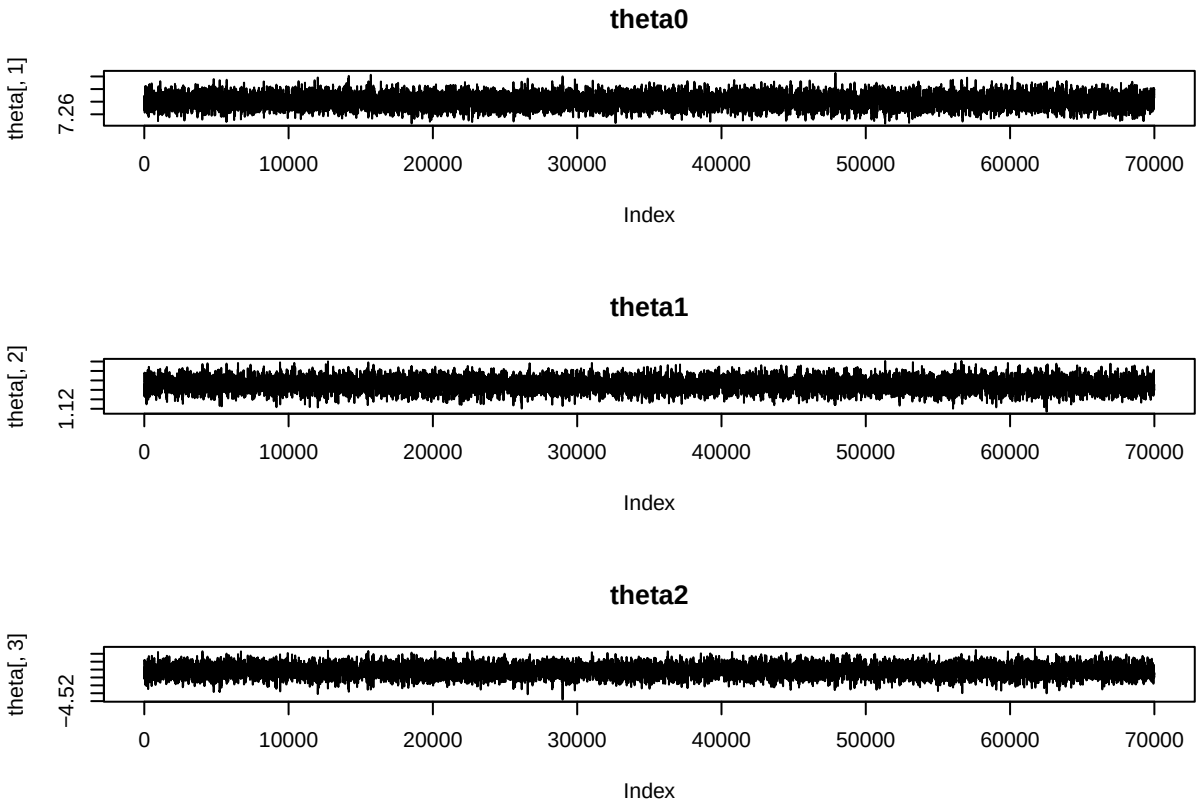
Here we reuse the algorithm from Q3 but this time we update the vector of theta's instead of the component one at a time. We will use a Normal trivariate distribution for the proposal and use the variance-covariance matrix computed with the MLE as follow : $\Sigma = \sigma^2 * \mathbf{J}(\hat{\theta})^{-1}$ where we use a $\sigma = 1.85$. We will use 70000 iterations (Appendix: Code Q4a-1).

We get an acceptance rate of

```
[1] 0.207503
```

And the plot of the different θ's are here :

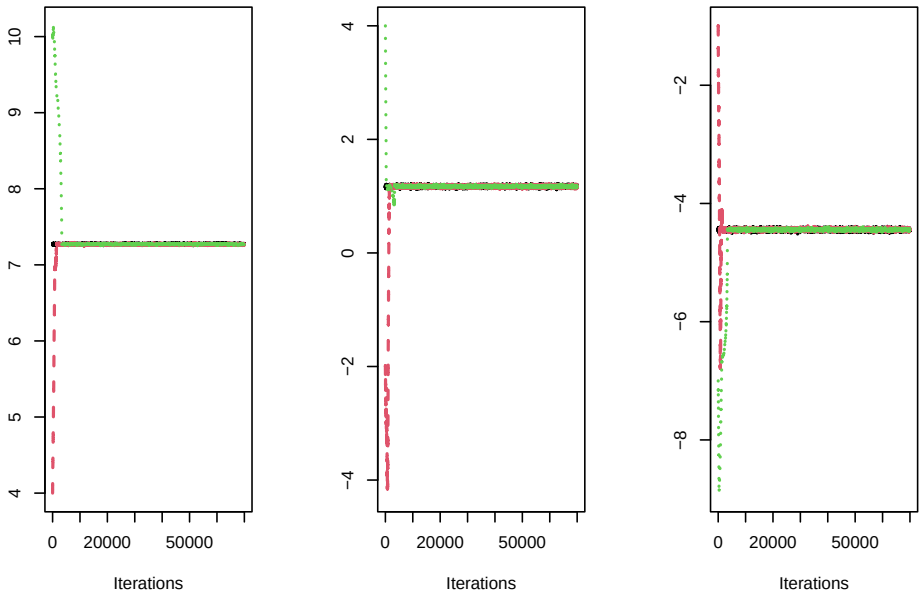
	x
2.5%	1423.099
50%	1436.855
97.5%	1452.741



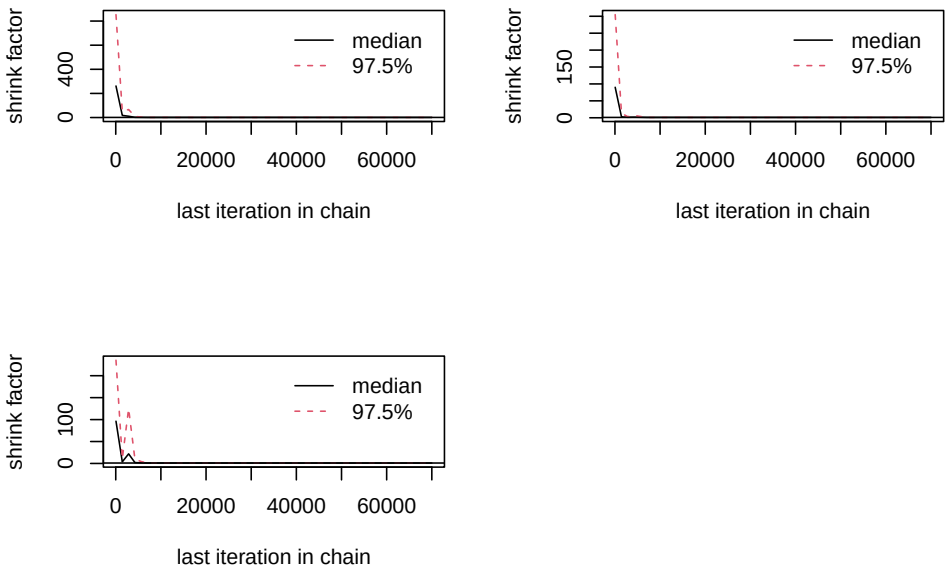
We can see that the standard deviation needed to achieve an acceptance rate of 0.2 is much larger than the componentwise Metropolis. It's because this time we take advantage of the covariance matrix of the MLE computed earlier and by taking a look at the matrix, the variances are really low (2.6e-05, 1.7e-04, 2.07e-04).

For the same reason than earlier, we will evaluate the convergence of our algorithm. We will use two different vector proposals : (4,-2,-1) and (10,4,-7) with respect to the signs of the MLE (Appendix: Code Q4a-2).

Here are the different traceplots :



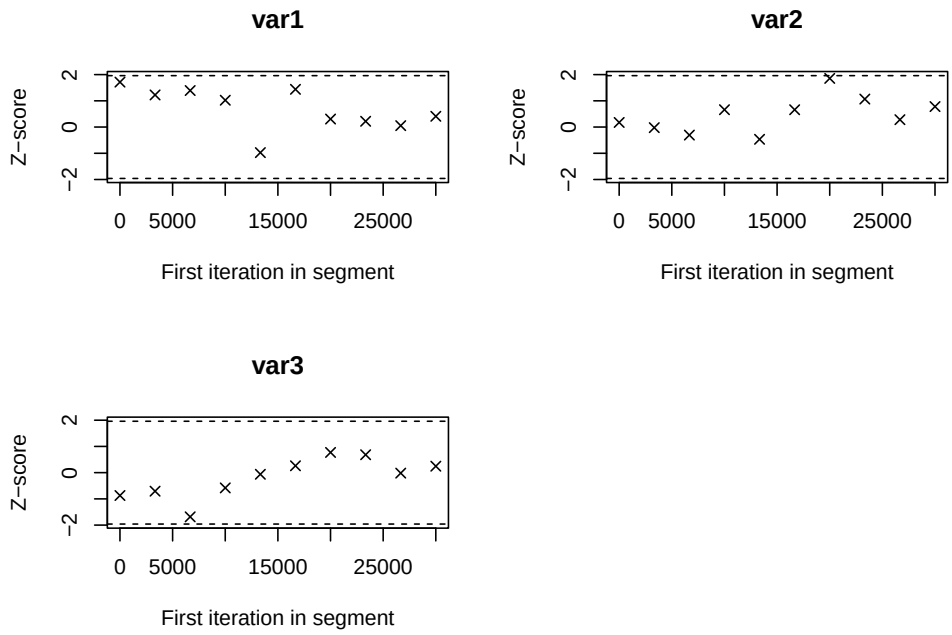
And the Gelman plots :



Convergence occur really fast and also here a burning of 10000 seems to be a suitable choice. We now compute the effective size of our sample and the different convergence diagnostics (Appendix: Code Q4a-3).

The geweke plots and effective size samples :

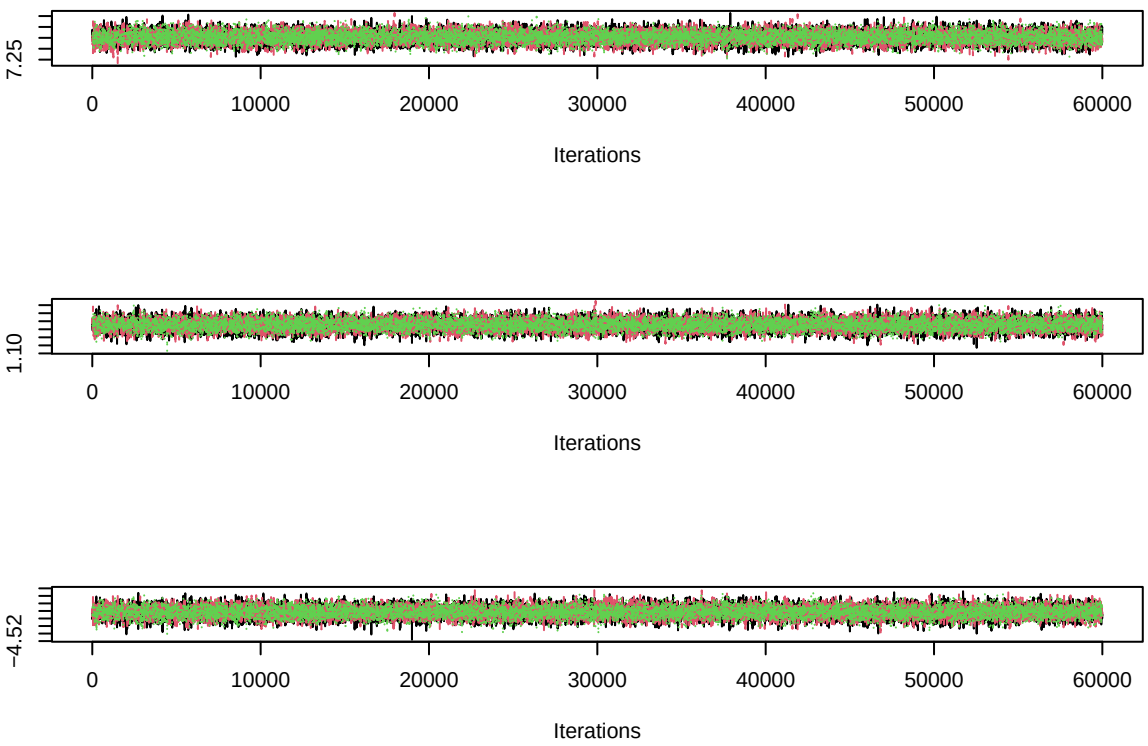
```
    var1     var2     var3
5538.974 5112.344 5242.443
```

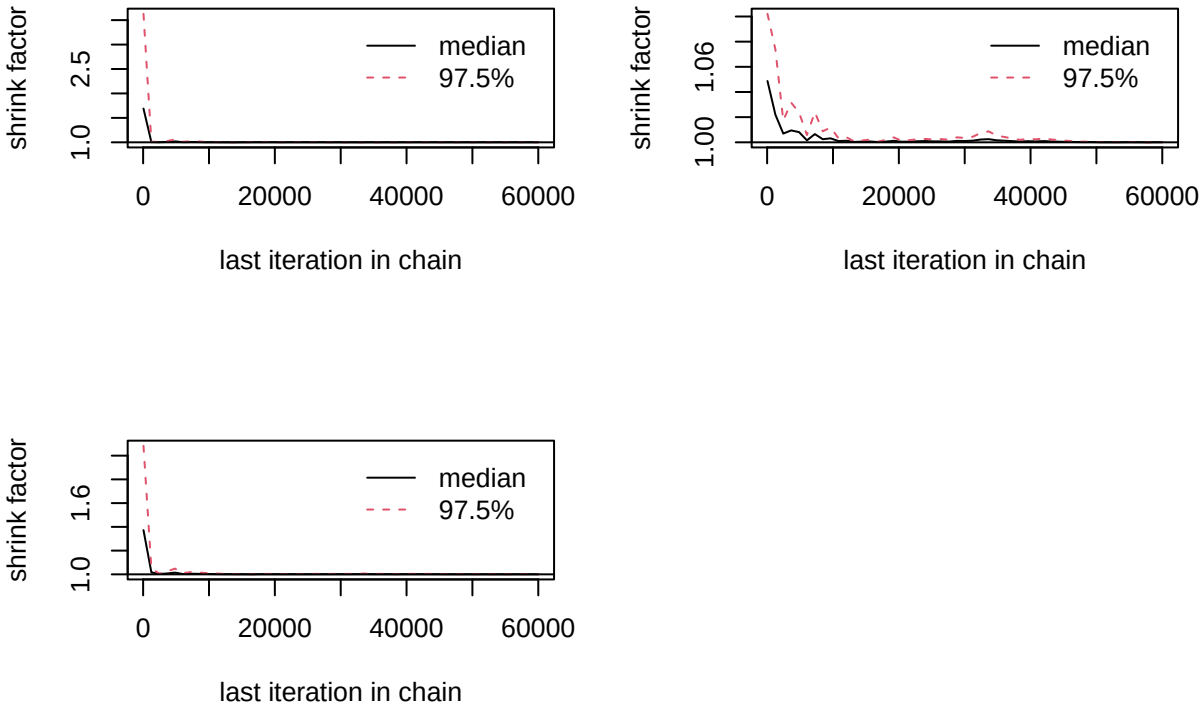


The effective samples are really high compare to the componentwise metropolis !

The traceplot, Gelman $\hat{R}$ and Gelman plots of the thinned samples:

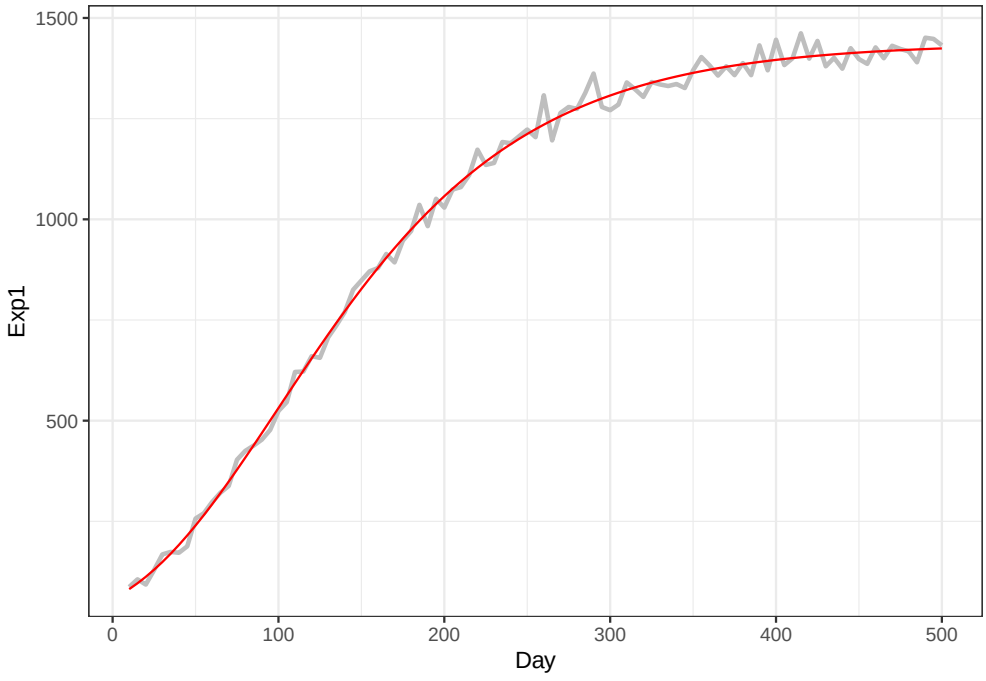
```
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1          1
[2,]          1          1
[3,]          1          1

Multivariate psrf

1
```


7.25
0
10000
20000
30000
40000
50000
60000
Iterations
1.10
0
10000
20000
30000
40000
50000
60000
Iterations
−4.52
0
10000
20000
30000
40000
50000
60000
Iterations



The gelman statistics are all really close to 1, indicating that convergence may have occur.

Compare the observed count data series for Experiment 1 with the fitted curve for $\mu(t)$ using our sample. Here is the plot comparing the date and the curve obtained with our sample (Appendix: Code 4a-4):



Seems that our sample of θ represents quite well the true value of θ.

Provide point estimates and 95% credible regions for $\beta_0, \beta_1, \beta_2$.

To do so, we will use the sample of θ to compute the α's and then the β's by the formulas calculated earlier (Appendix: Code Q4a-5).

	2.5%	50%	97.5%
Beta0	52.3340184	57.0587687	62.3326027
Beta1	0.0360290	0.0379463	0.0399230
Beta2	0.0114253	0.0117623	0.0120958

The values seem really close to the componentwise metropolis.

Provide a point estimate and a 95% credible region for the expected number of cancer cells at time $t = 0$ and $t = +\infty$ for Experiment 1 (Appendix: Code Q4a-6).

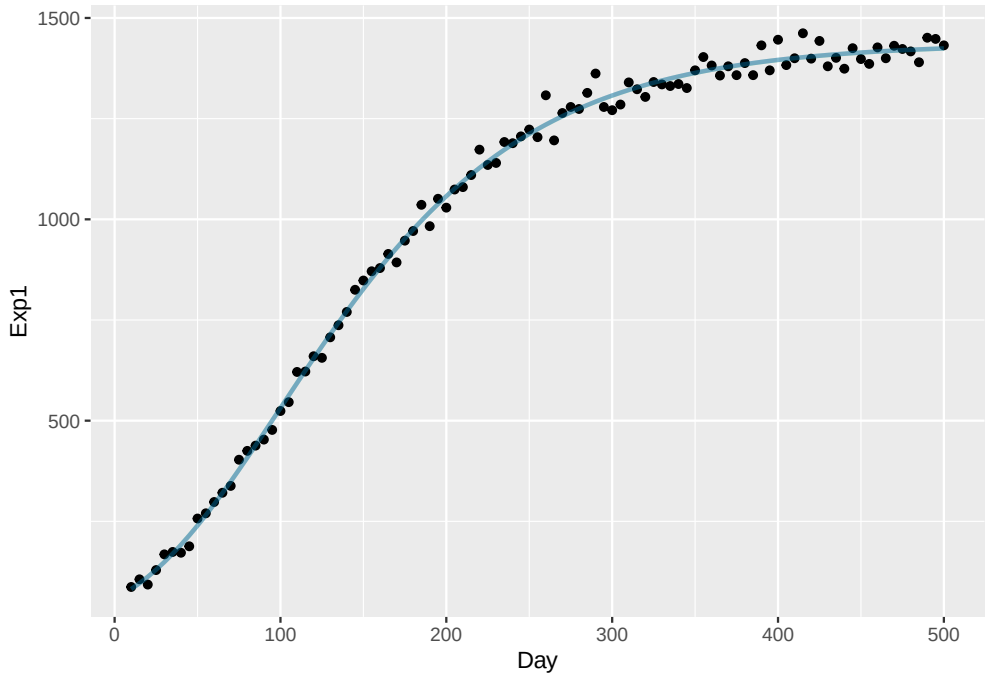
```
    2.5%      50%    97.5%
52.33402 57.05877 62.33260

    2.5%      50%    97.5%
1423.423 1437.171 1451.726

    2.5%      50%    97.5%
1423.423 1437.171 1451.726
```

Again, the interval is equal to those of α_0.

4 b)

Compare your results with the previous ones.

First, let's take a look at the predicted $\mu(t)$ for the two models (Appendix: Code Q4b-1):



We can't see the different lines which proves that they are really close.

We can take a look at the interval created by the two models (Appendix: Code Q4b-2):

Beta	Componentwise model			Vector proposal model		
	2.5%	50%	97.5%	2.5%	50%	97.5%
Beta0	52.1191389	57.1366311	62.3908190	52.3340184	57.0587687	62.3326027
Beta1	0.0360160	0.0379389	0.0399494	0.0360290	0.0379463	0.0399230
Beta2	0.0114241	0.0117650	0.0120985	0.0114253	0.0117623	0.0120958

The β's values are really close but we can see that the interval for the vector proposal model is slightly closer.

For the predictions (Appendix: Code Q4b-3):

	2.5%	50%	97.5%
Componentwise Model Prediction	52.11914	57.13663	62.39082
Vector Model Prediction	52.33402	57.05877	62.33260

	2.5%	50%	97.5%
Componentwise Model Prediction	1423.099	1436.855	1452.741
Vector Model Prediction	1423.423	1437.171	1451.726

As before, the values are really close to each other.

Eventually, the only difference is in the Metropolis algorithm itself. The second model is more efficient than the first one because we have convergence more rapidly, the effective sample size is larger and we don't have to have a very small proposal variance to construct an efficient model ($sd = 0.006$ against $sd = 1.1$).

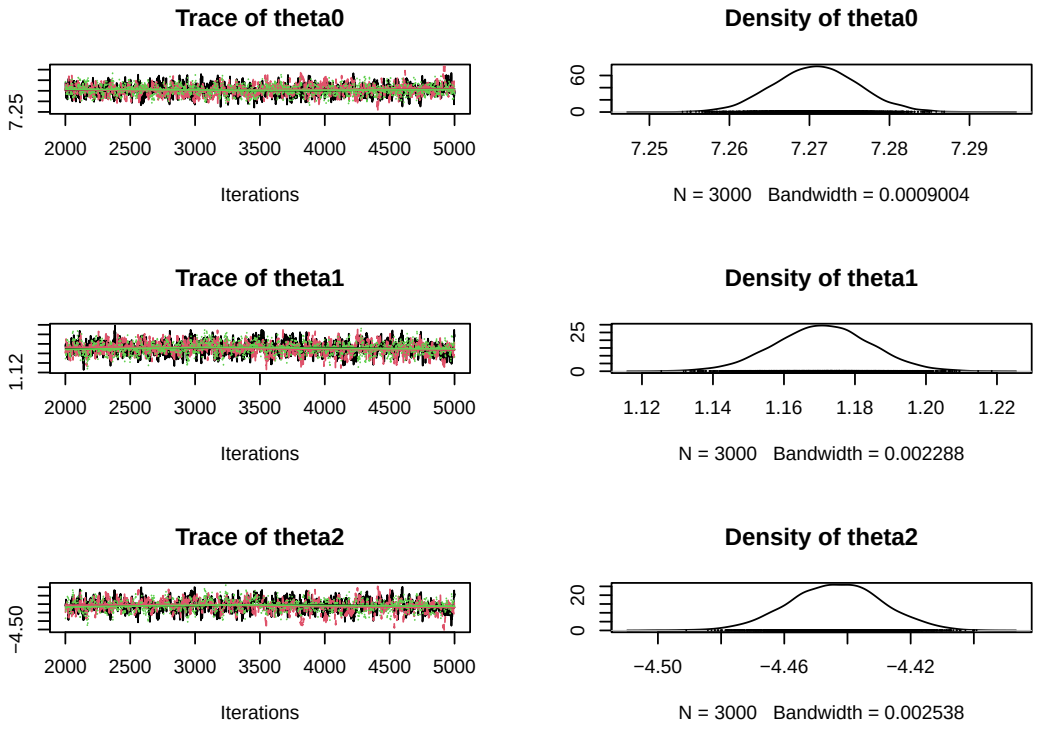
As a conclusion, the Metropolis algorithm construct with a vector proposal distribution of θ is more efficient than the componentwise way.

Question 5

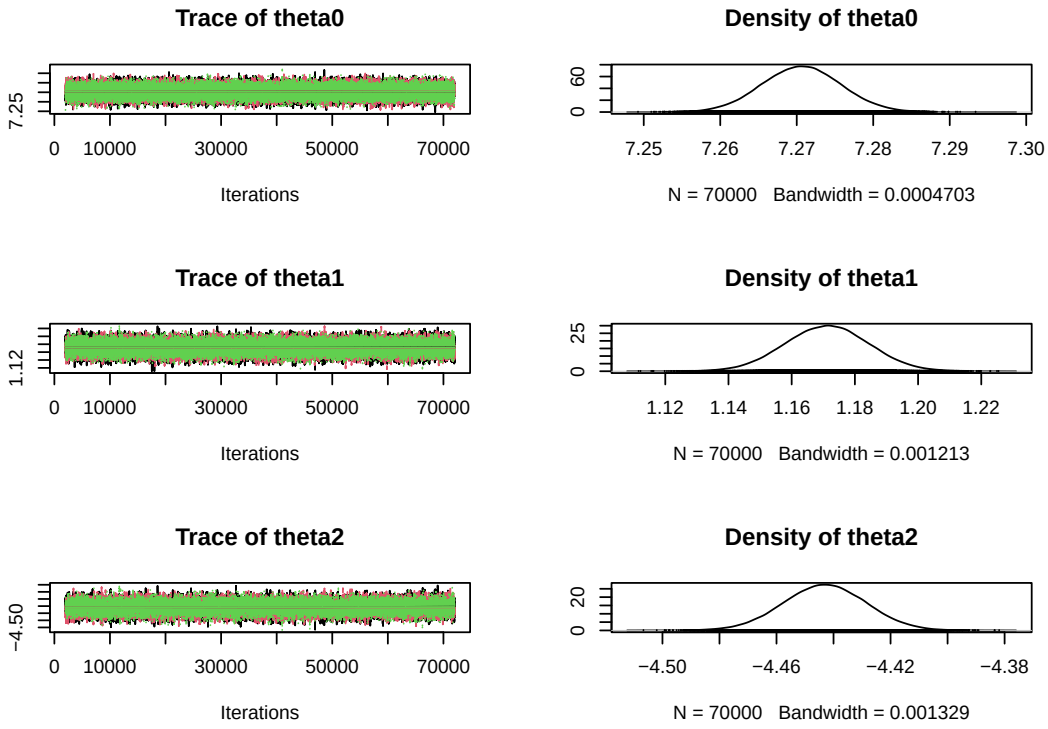
RJAGS, is a powerful software package that acts as a bridge between R and JAGS (Just Another Gibbs Sampler). JAGS is specialized in analyzing Bayesian analyse through Markov Chain Monte Carlo (MCMC) simulations. We will use RJAGS to sample from the posterior distribution $p(\theta|D_1)$ and compare the results obtained through RJAGS with the outcomes of the Metropolis algorithm implemented manually in previous questions.

5 a)

To start, our initial step involves creating the model and subsequently plotting the outcomes of the chains to obtain the distributions for each θ. This crucial process enables us to visually examine and analyze the variability and attributes of the parameter estimates. To accomplish this, we will set the initial value of each θ to zero. Additionally, we will employ five chains and conduct 3000 iterations (Appendix: Code Q5a-1).



Upon observation, we notice that the chains for θ_0 exhibit two distinct possibilities, suggesting a bimodal distribution with modes centered around 6.93 and 7.27. This bimodality indicates the presence of multiple modes in the posterior distribution, implying different plausible explanations or parameter configurations that can adequately explain the observed data. To further investigate this phenomenon, we will explore the impact of varying the initial value for θ_0 while utilizing the values obtained from the initial analysis. Additionally, we will enhance the precision of our analysis by increasing the number of iterations to 70,000 as in question 3 (Appendix: Code Q5a-2):



Now, upon further analysis, we observe a well-defined binomial distribution shape. After examining the summary of our MCMC results, we obtain specific values for each θ. These values provide us the estimated parameters and their corresponding uncertainties.

```
Iterations = 2001:72000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 70000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean       SD  Naive SE Time-series SE
theta0  7.271 0.005147 1.123e-05      3.448e-05
theta1  1.171 0.013278 2.897e-05      9.776e-05
theta2 -4.443 0.014539 3.173e-05      1.234e-04

2. Quantiles for each variable:

         2.5%    25%    50%    75%  97.5%
theta0  7.261  7.267  7.271  7.274  7.281
theta1  1.145  1.162  1.171  1.180  1.197
theta2 -4.472 -4.453 -4.443 -4.433 -4.415
```

5 b)

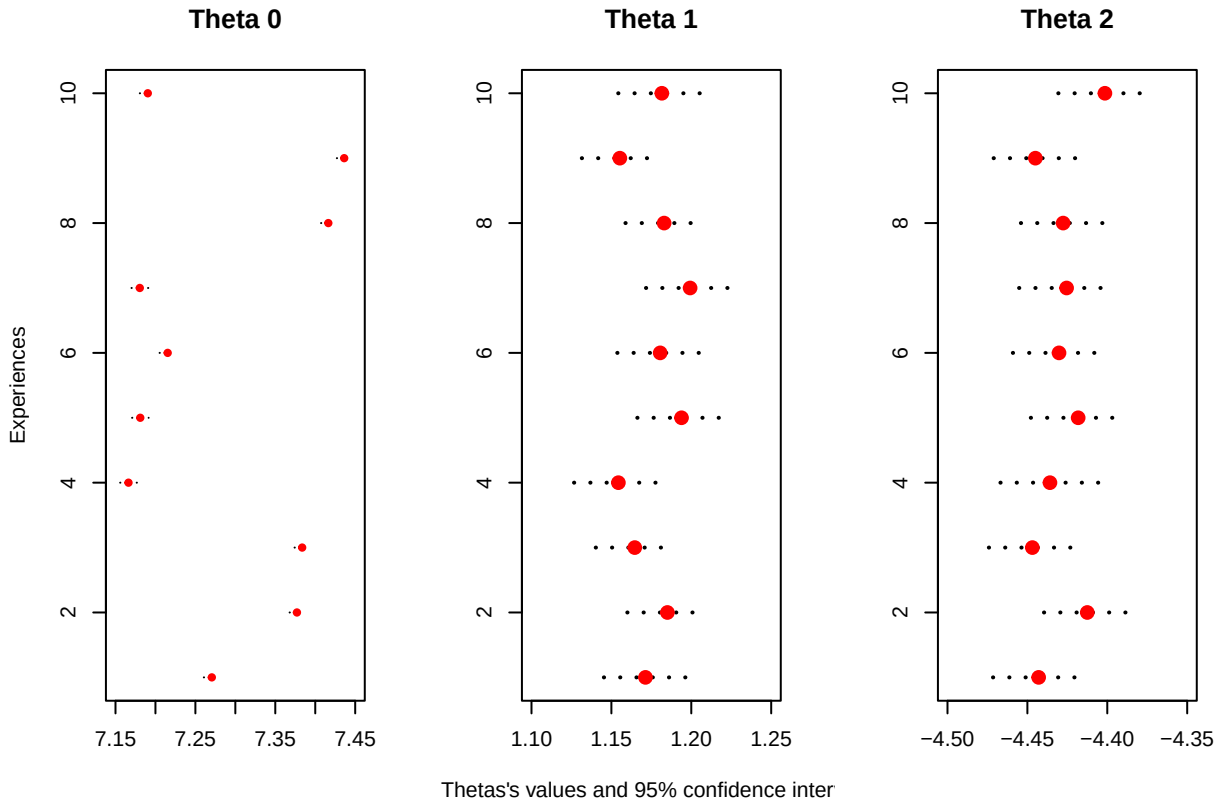
Utilizing our results, we employ the estimated parameters to calculate the number of cells at $t = \infty$. By leveraging the information obtained from the MCMC analysis, we can make an informed prediction regarding the long-term cell count. Based on the 95% confidence interval, the estimated number of cells at $t = \infty$ is 1438.654 (Appendix: Code Q5b).

```
    2.5%      50%    97.5%
1429.571 1429.571 1429.571
```

Question 6

To enhance our analysis, we employ a hierarchical model to calculate individual θ values for each experiment. This approach allows us to incorporate additional information and account for the inherent variability across experiments. By comparing the results from the hierarchical model with the previous analyses, we can assess the impact of this modeling improvement on our parameter estimates and evaluate any differences. To initiate our calculations, as in the previous question we assign the following initial values for each respective θ: 7, 1, and -4. Our analysis involves utilizing three chains and conducting 70,000 iterations (Appendix: Code Q6a).

6 a)



Upon analysis, we observe a consistent behavior for θ_1 and θ_2 across the experiments, which is expected. However, we are surprised to find again variations in the values of θ_0 among the experiments. This finding

challenges our previous resolution of a stable binomial distribution and raises questions again about a multimodale distribution in the data and suggest that our first analyse was not merely a result of convergence issues in the MCMC chains.

6 b)

Next, we will proceed with a comparison of our results specifically for experiment 1 (Appendix: Code Q6b).

	CI 2.5%	Median	CI 97.5%
Theta0 Q.5	7.260609	7.270629	7.280817
Theta0 Q.6	7.260529	7.270552	7.280791

	CI 2.5%	Median	CI 97.5%
Theta1 Q.5	1.145186	1.171218	1.197144
Theta1 Q.6	1.159945	1.185059	1.209958

	CI 2.5%	Median	CI 97.5%
Theta2 Q.5	-4.471771	-4.443075	-4.414707
Theta2 Q.6	-4.474133	-4.446887	-4.420038

For experiment 1, while comparing the results, we observe differences in values beyond the third decimal digit.

6 c)

Using the estimated parameters, we calculate the number of cells at $t = \infty$. Upon comparison with the outcome from question 5, we find that the results are, as expected, very close, indicating a finite value for convergence at approximately 1400 cells (Appendix: Code Q6c).

	CI 2.5%	Median	CI 97.5%
Final number of cells Q.5	1435.405	1435.405	1435.405
Final number of cells Q.6	1429.571	1429.571	1429.571

Conclusion

In conclusion, this project has significantly improved our understanding of the Bayesian approach. By utilizing MCMC and carefully selecting hyperparameters, we gained valuable insights into our model's behavior, enabling further investigation on coeficients,θ, used . RJAGS proved to be a useful package, simplifying the implementation of the Metropolis algorithm. Overall, this project has deepened our knowledge of Bayesian methods and equipped us with practical tools for future endeavors in this field.

Appendix

Code I.1

```
t <- cell$day
Di = cell$Exp1

model1 <- function(beta){
  beta0 <- beta[1]
  beta1 <- beta[2]
  beta2 <- beta[3]
  beta0 * exp(beta1 / (beta2) * (1 - exp(-beta2 * t)))
}

model2  = function(alpha){
  alpha0 <- alpha[1]
  alpha1 <- alpha[2]
  alpha2 <- alpha[3]
  alpha0 * exp(-alpha1 * exp(-alpha2 * t))
}
```

Code Q2b

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

  mu = exp(theta0)*exp(-exp(theta1)*exp(-exp(theta2)*t))
  ll <- sum(dpois(Di,lambda = mu, log=TRUE))

  return(ll)
}
```

Code Q2c

```
logprior = function(theta){
  theta0 = theta[1]
  theta1 = theta[2]
  theta2 = theta[3]
  log(dunif(theta0, min = -1e6, max = 1e6) *
        dunif(theta1, min = -1e6, max = 1e6) *
        dunif(theta2, min = -1e6, max = 1e6))
}

logpost = function(theta){
   loglike(theta) + logprior(theta)
}
```

Code Q2d

```r
neglogpost = function(theta){
   -logpost(theta)
}

fit <- optim(par = c(1500, 2.9, 0.01) |> log(), fn = neglogpost, hessian = TRUE)

theta_hat <- fit$par
theta_hat

ggplot(data = cell, aes(x = day, y = Exp1)) +
  geom_point() +
    geom_line(aes(y = as.matrix(model2(theta_hat |> exp()))))

covmat <- solve(fit$hessian)
covmat
```

Code Q3a

```r
M = 70000
theta1 = theta2 = theta3 = numeric(M)

theta1[1] <- theta_hat[1]; theta2[1] <- theta_hat[2]; theta3[1] <- theta_hat[3] # Starting value (MLE)
sd.prop1 = 0.07 ; sd.prop2 = .13 ; sd.prop3 = 0.026 ; n.accept = 0

for (i in 2:M) {

            theta <- c(theta1[i-1], theta2[i-1], theta3[i-1])

  theta1.prop <- theta1[i-1] + rnorm(1, 0, sd = sd.prop1) # sample theta 1
    theta.prop <- c(theta1.prop, theta2[i-1], theta3[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta1[i] <- theta1.prop ; n.accept = n.accept + 1
} else theta1[i] <- theta1[i-1]

  theta2.prop <- theta2[i-1] + rnorm(1, 0, sd = sd.prop2) # sample theta 2
    theta.prop <- c(theta1[i], theta2.prop, theta3[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta2[i] <- theta2.prop ; n.accept = n.accept + 1
} else theta2[i] <- theta2[i-1]

  theta3.prop <- theta3[i-1] + rnorm(1, 0, sd = sd.prop3) # sample theta 3
    theta.prop <- c(theta1[i], theta2[i], theta3.prop)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta3[i] <- theta3.prop ; n.accept = n.accept + 1
  } else theta3[i] <- theta3[i-1]
```

```
}

acceptance.rate = n.accept/(M-1)
acceptance.rate

plot(theta1, type = "l")
plot(theta2, type = "l")
plot(theta3, type = "l")
```

Code Q3b-1

```
# Second chain with theta = c(0,0,0)

theta1.1 = theta2.1 = theta3.1 = numeric(M)

theta1.1[1] <- 0; theta2.1[1] <- 0; theta3.1[1] <- 0 # Starting value
sd.prop1 = 0.07 ; sd.prop2 = .13 ; sd.prop3 = 0.026 ; n.accept = 0

for (i in 2:M) {

            theta <- c(theta1.1[i-1], theta2.1[i-1], theta3.1[i-1])

  theta1.prop <- theta1.1[i-1] + rnorm(1, 0, sd = sd.prop1) # sample theta 1
    theta.prop <- c(theta1.prop, theta2.1[i-1], theta3.1[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta1.1[i] <- theta1.prop ; n.accept = n.accept + 1
} else theta1.1[i] <- theta1.1[i-1]

  theta2.prop <- theta2.1[i-1] + rnorm(1, 0, sd = sd.prop2) # sample theta 2
    theta.prop <- c(theta1.1[i], theta2.prop, theta3.1[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta2.1[i] <- theta2.prop ; n.accept = n.accept + 1
} else theta2.1[i] <- theta2.1[i-1]

  theta3.prop <- theta3.1[i-1] + rnorm(1, 0, sd = sd.prop3) # sample theta 3
    theta.prop <- c(theta1.1[i], theta2.1[i], theta3.prop)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta3.1[i] <- theta3.prop ; n.accept = n.accept + 1
  } else theta3.1[i] <- theta3.1[i-1]


}
```

```
theta1.2 = theta2.2 = theta3.2 = numeric(M)

theta1.2[1] <- 8.5; theta2.2[1] <- 3.5; theta3.2[1] <- -6.5
# Starting value (signs with respect to the MLE)
sd.prop1 = 0.07 ; sd.prop2 = .13 ; sd.prop3 = 0.026 ; n.accept = 0

for (i in 2:M) {

            theta <- c(theta1.2[i-1], theta2.2[i-1], theta3.2[i-1])

  theta1.prop <- theta1.2[i-1] + rnorm(1, 0, sd = sd.prop1) # sample theta 1
    theta.prop <- c(theta1.prop, theta2.2[i-1], theta3.2[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta1.2[i] <- theta1.prop ; n.accept = n.accept + 1
} else theta1.2[i] <- theta1.2[i-1]

  theta2.prop <- theta2.2[i-1] + rnorm(1, 0, sd = sd.prop2) # sample theta 2
    theta.prop <- c(theta1.2[i], theta2.prop, theta3.2[i-1])
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta2.2[i] <- theta2.prop ; n.accept = n.accept + 1
} else theta2.2[i] <- theta2.2[i-1]

  theta3.prop <- theta3.2[i-1] + rnorm(1, 0, sd = sd.prop3) # sample theta 3
    theta.prop <- c(theta1.2[i], theta2.2[i], theta3.prop)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta)))

  accept = (prob >= runif(1))
  if (accept) {
    theta3.2[i] <- theta3.prop ; n.accept = n.accept + 1
  } else theta3.2[i] <- theta3.2[i-1]


}

traceplot(list(mcmc(theta1), mcmc(theta1.1), mcmc(theta1.2)))
traceplot(list(mcmc(theta2), mcmc(theta2.1), mcmc(theta2.2)))
traceplot(list(mcmc(theta3), mcmc(theta3.1), mcmc(theta3.2)))

gelman.plot(list(mcmc(theta1), mcmc(theta1.1), mcmc(theta1.2)))
gelman.plot(list(mcmc(theta2), mcmc(theta2.1), mcmc(theta2.2)))
gelman.plot(list(mcmc(theta3), mcmc(theta3.1), mcmc(theta3.2)))
```

Code Q3b-2

```
# burnin of 10000
wburn <- 10000:M
```

```
geweke.plot(mcmc(theta1[wburn]), nbins = 9)
geweke.plot(mcmc(theta2[wburn]), nbins = 9)
geweke.plot(mcmc(theta3[wburn]), nbins = 9)

effectiveSize(mcmc(theta1[wburn]))
effectiveSize(mcmc(theta2[wburn]))
effectiveSize(mcmc(theta3[wburn]))

traceplot(list(mcmc(theta1[wburn]), mcmc(theta1.1[wburn]), mcmc(theta1.2[wburn])))
traceplot(list(mcmc(theta2[wburn]), mcmc(theta2.1[wburn]), mcmc(theta2.2[wburn])))
traceplot(list(mcmc(theta3[wburn]), mcmc(theta3.1[wburn]), mcmc(theta3.2[wburn])))

gelman.diag(list(mcmc(theta1[wburn]), mcmc(theta1.1[wburn]), mcmc(theta1.2[wburn])))
gelman.diag(list(mcmc(theta2[wburn]), mcmc(theta2.1[wburn]), mcmc(theta2.2[wburn])))
gelman.diag(list(mcmc(theta3[wburn]), mcmc(theta3.1[wburn]), mcmc(theta3.2[wburn])))

gelman.plot(list(mcmc(theta1[wburn]), mcmc(theta1.1[wburn]), mcmc(theta1.2[wburn])))
gelman.plot(list(mcmc(theta2[wburn]), mcmc(theta2.1[wburn]), mcmc(theta2.2[wburn])))
gelman.plot(list(mcmc(theta3[wburn]), mcmc(theta3.1[wburn]), mcmc(theta3.2[wburn])))
```

Code Q3c

```
Observed <- cell$Exp1

alpha0 = exp(theta1[wburn])
alpha1 = exp(theta2[wburn])
alpha2 = exp(theta3[wburn])

fitcurve3 <- c()

for (i in 1:length(t)) {

  fit <- alpha0 * exp(-alpha1 * exp(-alpha2 * t[i]))
  fitcurve3[i] <- mean(fit)
}


p3 <- data.frame(Day = cell$day, Exp1 = Observed, estimM1 = fitcurve3)

ggplot(p3) +
  aes(x = Day) +
    geom_line(aes(y = Exp1), col = "gray", linewidth=1) +
    geom_line(aes(y = estimM1), col = "red")
```

Code Q3d

```
beta0_3 = alpha0 * exp(-alpha1)
beta1_3 = alpha1 * alpha2
beta2_3 = alpha2

rbind(Beta0 = quantile(beta0_3, p = c(0.025,0.5,0.975)),
      Beta1 = quantile(beta1_3, p = c(0.025,0.5,0.975)),
      Beta2 = quantile(beta2_3, p = c(0.025,0.5,0.975))) |>
```

```
  kbl() |>
    kable_styling()
```

Code Q3e

```
# at t=0

pred1_3 <- alpha0 * exp(-alpha1 * exp(-alpha2 * 0))
quantile(pred1_3, p = c(0.025,0.5,0.975))|> kbl() |> kable_styling()

#at t =+inf

pred2_3 <- alpha0 * exp(-alpha1 * exp(-alpha2 * Inf))
quantile(pred2_3, p = c(0.025,0.5,0.975)) |> kbl() |> kable_styling()

quantile(alpha0, p = c(0.025,0.5,0.975)) |> kbl() |> kable_styling()
```

Code Q4a-1

```
M = 70000
theta = matrix(nrow = M, ncol=3)

theta[1,] <- theta_hat # Starting value (MLE)
sd.prop = 1.85 ; n.accept = 0

for (i in 2:M) {


  theta.prop <- theta[i-1,] + rmnorm(1, c(0,0,0),varcov = sd.prop^2 * covmat)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta[i-1,])))

  accept = (prob >= runif(1))
  if (accept) {
    theta[i,] <- theta.prop
    n.accept = n.accept + 1
  }
  else theta[i,] <- theta[i-1,]

}

acceptance.rate = n.accept/(M-1)
acceptance.rate

plot(theta[,1], type = "l")
plot(theta[,2], type = "l")
plot(theta[,3], type = "l")
```

Code Q4a-2

```
# Second chain with theta = c(4,-2,-1)

theta.chain2 = matrix(nrow = M, ncol=3)
```

```
theta.chain2[1,] <- c(4,-2,-1) # Starting value (MLE)
sd.prop = 1.85 ; n.accept = 0

for (i in 2:M) {


  theta.prop <- theta.chain2[i-1,] + rmnorm(1, c(0,0,0),varcov = sd.prop^2 * covmat)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta.chain2[i-1,])))

  accept = (prob >= runif(1))
  if (accept) {
    theta.chain2[i,] <- theta.prop
    n.accept = n.accept + 1
  }
  else theta.chain2[i,] <- theta.chain2[i-1,]

}

# Third chain with theta = c(10,4,-7)

theta.chain3 = matrix(nrow = M, ncol=3)

theta.chain3[1,] <- c(10,4,-7) # Starting value (MLE)
sd.prop = 1.85 ; n.accept = 0

for (i in 2:M) {


  theta.prop <- theta.chain3[i-1,] + rmnorm(1, c(0,0,0),varcov = sd.prop^2 * covmat)
    prob = min(1, exp(logpost(theta.prop)-logpost(theta.chain3[i-1,])))

  accept = (prob >= runif(1))
  if (accept) {
    theta.chain3[i,] <- theta.prop
    n.accept = n.accept + 1
  }
  else theta.chain3[i,] <- theta.chain3[i-1,]

}

traceplot(list(mcmc(theta), mcmc(theta.chain2), mcmc(theta.chain3)))

gelman.plot(list(mcmc(theta), mcmc(theta.chain2), mcmc(theta.chain3)))
```

Code Q4a-3

```
# burnin of 10000
wburn <- 10000:M

geweke.plot(mcmc(theta[wburn,]), nbins = 10)

effectiveSize(mcmc(theta[wburn,]))
```

```
traceplot(list(mcmc(theta[wburn,]), mcmc(theta.chain2[wburn,]), mcmc(theta.chain3[wburn,])))

gelman.diag(list(mcmc(theta[wburn,]), mcmc(theta.chain2[wburn,]), mcmc(theta.chain3[wburn,])))

gelman.plot(list(mcmc(theta[wburn,]), mcmc(theta.chain2[wburn,]), mcmc(theta.chain3[wburn,])))
```

Code Q4a-4

```
Observed <- cell$Exp1

alpha0 = exp(theta[wburn,1])
alpha1 = exp(theta[wburn,2])
alpha2 = exp(theta[wburn,3])

fitcurve4 <- c()

for (i in 1:length(t)) {

  fit <- alpha0 * exp(-alpha1 * exp(-alpha2 * t[i]))
  fitcurve4[i] <- mean(fit)
}

p4 <- data.frame(Day = cell$day, Exp1 = Observed, estimM2 = fitcurve4)

ggplot(p4) +
  aes(x = Day) +
    geom_line(aes(y = Exp1), col = "gray", linewidth=1) +
    geom_line(aes(y = estimM2), col = "red")
```

Code Q4a-5

```
beta0_4 = alpha0 * exp(-alpha1)
beta1_4 = alpha1 * alpha2
beta2_4 = alpha2

rbind(Beta0 = quantile(beta0_4, p = c(0.025,0.5,0.975)),
      Beta1 = quantile(beta1_4, p = c(0.025,0.5,0.975)),
      Beta2 = quantile(beta2_4, p = c(0.025,0.5,0.975))) |>
  kbl() |>
    kable_styling()
```

Code Q4a-6

```
# at t=0

pred1_4 <- alpha0 * exp(-alpha1 * exp(-alpha2 * 0))
quantile(pred1_4, p = c(0.025,0.5,0.975))

#at t =+inf

pred2_4 <- alpha0 * exp(-alpha1 * exp(-alpha2 * Inf))
quantile(pred2_4, p = c(0.025,0.5,0.975))
```

```
quantile(alpha0, p = c(0.025,0.5,0.975))
```

Code Q4b-1

```
ggplot(p3) +
  aes(x = Day) +
    geom_point(aes(y = Exp1), size=1.5) +
      geom_line(aes(y = estimM1), col= "green", linewidth = 1, alpha = 0.3) +
      geom_line(data = p4, mapping = aes(y=estimM2), col="blue", linewidth = 1, alpha = 0.3)
```

Code Q4b-2

```
tab4 <- rbind(Beta0 = quantile(beta0_4, p = c(0.025,0.5,0.975)),
              Beta1 = quantile(beta1_4, p = c(0.025,0.5,0.975)),
              Beta2 = quantile(beta2_4, p = c(0.025,0.5,0.975)))

tab3 <- rbind(Beta0 = quantile(beta0_3, p = c(0.025,0.5,0.975)),
              Beta1 = quantile(beta1_3, p = c(0.025,0.5,0.975)),
              Beta2 = quantile(beta2_3, p = c(0.025,0.5,0.975)))
tab <- cbind(tab3, tab4)
tab |>
  kbl() |>
    kable_minimal() |>
      add_header_above(header = c("Beta" = 1,"Componentwise model"=3, "Vector proposal model"=3))
```

Code Q4b-3

```
#at t=0
rbind("Componentwise Model Prediction" = quantile(pred1_3, p = c(0.025,0.5,0.975)),
      "Vector Model Prediction" = quantile(pred1_4, p = c(0.025,0.5,0.975))) |>
  kbl() |>
    kable_styling()

# at t = Inf

rbind("Componentwise Model Prediction" = quantile(pred2_3, p = c(0.025,0.5,0.975)),
      "Vector Model Prediction" = quantile(pred2_4, p = c(0.025,0.5,0.975))) |>
  kbl() |>
    kable_styling()
```

Code Q5a-1

```
# Model description
mymodel = function(){
  for (i in 1:n){
    y[i] ~ dpois(mu[i])
    mu[i] <- exp(theta0) * exp(-exp(theta1)*exp(-exp(theta2)*t[i]))
  }
  theta0 ~ dunif(-1e5, 1e5)
  theta1 ~ dunif( -1e5, 1e5)
  theta2 ~ dunif(-1e5, 1e5)
```

```
}
model.file = "test.bug"
write.model(mymodel,model.file)

# Data and initial values for model parameters
y = cell$Exp1
t = cell$day
mydata = list(y=y, t = t, n=99)
inits = list( theta0 = 0, theta1 = 0, theta2 = 0)

# Model specification
foo = jags.model(file=model.file, data=mydata, inits=inits,
                 n.chains= 3)
update(foo, 1000)

# Generation of the chain
out = coda.samples(model=foo,
                   variable.names=c("theta0", "theta1", "theta2"),
                   n.iter=3000)
out.matrix = as.matrix(out)
# Inspection of the generated chains
plot(out)
#summary(out)
#HPDinterval(out)

out.question_5 = out.matrix
```

Code Q5a-2

```
# Model description
mymodel = function(){
  for (i in 1:n){
    y[i] ~ dpois(mu[i])
    mu[i] <- exp(theta0) * exp(-exp(theta1)*exp(-exp(theta2)*t[i]))
  }
  theta0 ~ dunif(-1e5, 1e5)
  theta1 ~ dunif( -1e5, 1e5)
  theta2 ~ dunif(-1e5, 1e5)
}
model.file = "test.bug"
write.model(mymodel,model.file)

# Data and initial values for model parameters
y = cell$Exp1
t = cell$day
mydata = list(y=y, t = t, n=99)
inits = list( theta0 = 0, theta1 = 0, theta2 = 0)

# Model specification
foo = jags.model(file=model.file, data=mydata, inits=inits,
                 n.chains= 3)
update(foo, 1000)
```

```
# Generation of the chain
out = coda.samples(model=foo,
                   variable.names=c("theta0", "theta1", "theta2"),
                   n.iter=70000)
out.matrix = as.matrix(out)
# Inspection of the generated chains
plot(out)
#summary(out)
#HPDinterval(out)

out.question_5 = out.matrix

summary(out)
```

Code Q5b

```
theta0 = out.matrix[1]
theta1 = out.matrix[2]
theta2 = out.matrix[3]


alpha0 = exp(theta0)
alpha1 = exp(theta1)
alpha2 = exp(theta2)


prediction <- alpha0 * exp(-alpha1 * exp(-alpha2 * Inf))
quantile(prediction, p = c(0.025,0.5,0.975))
```

Code Q6a

```
# Model description
mymodel = function(){
  for (i in 1:n){
    y.1[i] ~ dpois(mu.1[i])
    mu.1[i] <- exp(theta0.1) * exp(-exp(theta1.1)*exp(-exp(theta2.1)*t[i]))

    y.2[i] ~ dpois(mu.2[i])
    mu.2[i] <- exp(theta0.2) * exp(-exp(theta1.2)*exp(-exp(theta2.2)*t[i]))

    y.3[i] ~ dpois(mu.3[i])
    mu.3[i] <- exp(theta0.3) * exp(-exp(theta1.3)*exp(-exp(theta2.3)*t[i]))

    y.4[i] ~ dpois(mu.4[i])
    mu.4[i] <- exp(theta0.4) * exp(-exp(theta1.4)*exp(-exp(theta2.4)*t[i]))

    y.5[i] ~ dpois(mu.5[i])
    mu.5[i] <- exp(theta0.5) * exp(-exp(theta1.5)*exp(-exp(theta2.5)*t[i]))

    y.6[i] ~ dpois(mu.6[i])
    mu.6[i] <- exp(theta0.6) * exp(-exp(theta1.6)*exp(-exp(theta2.6)*t[i]))

    y.7[i] ~ dpois(mu.7[i])
```

```
    mu.7[i] <- exp(theta0.7) * exp(-exp(theta1.7)*exp(-exp(theta2.7)*t[i]))

    y.8[i] ~ dpois(mu.8[i])
    mu.8[i] <- exp(theta0.8) * exp(-exp(theta1.8)*exp(-exp(theta2.8)*t[i]))

    y.9[i] ~ dpois(mu.9[i])
    mu.9[i] <- exp(theta0.9) * exp(-exp(theta1.9)*exp(-exp(theta2.9)*t[i]))

    y.10[i] ~ dpois(mu.10[i])
    mu.10[i] <- exp(theta0.10) * exp(-exp(theta1.10)*exp(-exp(theta2.10)*t[i]))
  }

  theta0.1 ~ dunif(-1e5, 1e5)
  theta1.1 ~ dunif( -1e5, 1e5)
  theta2.1 ~ dunif(-1e5, 1e5)

  theta0.2 ~ dunif(-1e5, 1e5)
  theta1.2 ~ dunif( -1e5, 1e5)
  theta2.2 ~ dunif(-1e5, 1e5)

  theta0.3 ~ dunif(-1e5, 1e5)
  theta1.3 ~ dunif( -1e5, 1e5)
  theta2.3 ~ dunif(-1e5, 1e5)

  theta0.4 ~ dunif(-1e5, 1e5)
  theta1.4 ~ dunif( -1e5, 1e5)
  theta2.4 ~ dunif(-1e5, 1e5)

  theta0.5 ~ dunif(-1e5, 1e5)
  theta1.5 ~ dunif( -1e5, 1e5)
  theta2.5 ~ dunif(-1e5, 1e5)

  theta0.6 ~ dunif(-1e5, 1e5)
  theta1.6 ~ dunif( -1e5, 1e5)
  theta2.6 ~ dunif(-1e5, 1e5)

  theta0.7 ~ dunif(-1e5, 1e5)
  theta1.7 ~ dunif( -1e5, 1e5)
  theta2.7 ~ dunif(-1e5, 1e5)

  theta0.8 ~ dunif(-1e5, 1e5)
  theta1.8 ~ dunif( -1e5, 1e5)
  theta2.8 ~ dunif(-1e5, 1e5)

  theta0.9 ~ dunif(-1e5, 1e5)
  theta1.9 ~ dunif( -1e5, 1e5)
  theta2.9 ~ dunif(-1e5, 1e5)

  theta0.10 ~ dunif(-1e5, 1e5)
  theta1.10 ~ dunif( -1e5, 1e5)
  theta2.10 ~ dunif(-1e5, 1e5)
}
```

```
model.file = "test.bug"
write.model(mymodel,model.file)

# Data and initial values for model parameters
t = cell$day
mydata = list(y.1 = cell$Exp1, y.2 = cell$Exp2, y.3 = cell$Exp3, y.4 = cell$Exp4,
              y.5 = cell$Exp5, y.6 = cell$Exp6, y.7 = cell$Exp7, y.8 = cell$Exp8,
              y.9 = cell$Exp9, y.10 = cell$Exp10, t = t, n=99)

inits = list(theta0.1 = 7, theta1.1 = 1, theta2.1 = -4,
             theta0.2 = 7, theta1.2 = 1, theta2.2 = -4,
             theta0.3 = 7, theta1.3 = 1, theta2.3 = -4,
             theta0.4 = 7, theta1.4 = 1, theta2.4 = -4,
             theta0.5 = 7, theta1.5 = 1, theta2.5 = -4,
             theta0.6 = 7, theta1.6 = 1, theta2.6 = -4,
             theta0.7 = 7, theta1.7 = 1, theta2.7 = -4,
             theta0.8 = 7, theta1.8 = 1, theta2.8 = -4,
             theta0.9 = 7, theta1.9 = 1, theta2.9 = -4,
             theta0.10 = 7, theta1.10 = 1, theta2.10 = -4)

# Model specification
foo = jags.model(file=model.file, data=mydata, inits=inits,
                 n.chains = 3)

update(foo, 1000)

# Generation of the chain
out = coda.samples(model=foo,
                   variable.names=c("theta0.1", "theta1.1", "theta2.1",
                                    "theta0.2", "theta1.2", "theta2.2",
                                    "theta0.3", "theta1.3", "theta2.3",
                                    "theta0.4", "theta1.4", "theta2.4",
                                    "theta0.5", "theta1.5", "theta2.5",
                                    "theta0.6", "theta1.6", "theta2.6",
                                    "theta0.7", "theta1.7", "theta2.7",
                                    "theta0.8", "theta1.8", "theta2.8",
                                    "theta0.9", "theta1.9", "theta2.9",
                                    "theta0.10", "theta1.10","theta2.10"),n.iter=70000)

out.matrix = as.matrix(out)

theta0.i = out.matrix[,1:10]
theta1.i = out.matrix[,11:20]
theta2.i = out.matrix[,21:30]

## Plot of the posterior credible intervals
post.interv <- apply(theta0.i,2, quantile, p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)
par(mfrow=c(1,3))

plot(1:10, xlim = c(7.15,7.45), type="n", ylab="Experiences", main ="Theta 0", xlab = '')

for (j in 1:10){
```

```
  lines(c(post.interv[1,j],post.interv[3,j]),c(j,j), lty = 3, lwd = 1)
  points(post.interv[2,j],j,pch=20, cex = 1, col = 'red')
}

## Plot of the posterior credible intervals
post.interv <- apply(theta1.i,2,quantile,p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)

plot(1:10, xlim = c(1.1,1.25), type="n", main ="Theta 1", ylab = '', xlab = "Thetas's values and 95% co

for (j in 1:10){
  lines(c(post.interv[1,j],post.interv[3,j]),c(j,j), lty = 3, lwd = 2)
  points(post.interv[2,j],j,pch=20, cex = 2, col = 'red')
}

## Plot of the posterior credible intervals
post.interv <- apply(theta2.i,2,quantile,p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)

plot(1:10, xlim = c(-4.5,-4.35), type="n", main ="Theta 2", ylab = '', xlab = '')

for (j in 1:10){
  lines(c(post.interv[1,j],post.interv[3,j]),c(j,j), lty = 3, lwd = 2)
  points(post.interv[2,j],j,pch=20, cex = 2, col = 'red')
}
```

Code Q6b

```
theta0 = out.question_5[,1]
theta1 = out.question_5[,2]
theta2 = out.question_5[,3]

## Plot of the posterior credible intervals
post.interv <- apply(theta0.i,2,quantile,p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)
post.interv_5 <- quantile(theta0, p=c(.025,0.5,.975))

a = post.interv_5
b = post.interv[,1]
X = rbind(a,b)
Y = matrix(data = X, nrow = 2, ncol = 3)
Y = data.frame(Y)
colnames(Y) = c("CI 2.5%", 'Median', 'CI 97.5%')
rownames(Y) = c('Theta0 Q.5', 'Theta0 Q.6')
Y |> kable()

## Plot of the posterior credible intervals
post.interv <- apply(theta1.i,2,quantile,p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)
post.interv_5 <- quantile(theta1, p=c(.025,0.5,.975))

a = post.interv_5
b = post.interv[,2]
```

```
X = rbind(a,b)
Y = matrix(data = X, nrow = 2, ncol = 3)
Y = data.frame(Y)
colnames(Y) = c("CI 2.5%", 'Median', 'CI 97.5%')
rownames(Y) = c('Theta1 Q.5', 'Theta1 Q.6')
Y |> kable()

## Plot of the posterior credible intervals
post.interv <- apply(theta2.i,2,quantile,p=c(.025,0.5,.975))
#OR.obs = Event.Tr*(Total.Ctrl-Event.Ctrl)/Event.Ctrl/(Total.Tr-Event.Tr)
post.interv_5 <- quantile(theta2, p=c(.025,0.5,.975))

a = post.interv_5
b = post.interv[,3]
X = rbind(a,b)
Y = matrix(data = X, nrow = 2, ncol = 3)
Y = data.frame(Y)
colnames(Y) = c("CI 2.5%", 'Median', 'CI 97.5%')
rownames(Y) = c('Theta2 Q.5', 'Theta2 Q.6')
Y |> kable()
```

Code Q6c

```
theta0 = out.matrix[1]
theta1 = out.matrix[2]
theta2 = out.matrix[3]


alpha0 = exp(theta0)
alpha1 = exp(theta1)
alpha2 = exp(theta2)


prediction <- alpha0 * exp(-alpha1 * exp(-alpha2 * Inf))
a = quantile(prediction, p = c(0.025,0.5,0.975))


theta0 = out.question_5[1]
theta1 = out.question_5[2]
theta2 = out.question_5[3]


alpha0 = exp(theta0)
alpha1 = exp(theta1)
alpha2 = exp(theta2)


prediction <- alpha0 * exp(-alpha1 * exp(-alpha2 * Inf))
b = quantile(prediction, p = c(0.025,0.5,0.975))

X = rbind(a,b)
Y = matrix(data = X, nrow = 2, ncol = 3)
Y = data.frame(Y)
```

```
colnames(Y) = c("CI 2.5%", 'Median', 'CI 97.5%')
rownames(Y) = c('Final number of cells Q.5', 'Final number of cells Q.6')
Y |> kable()
```