STAT 578: Advanced Bayesian Modeling
Bahman Sheikh (NetID: bahmans2)

Data Analysis Report

Introduction

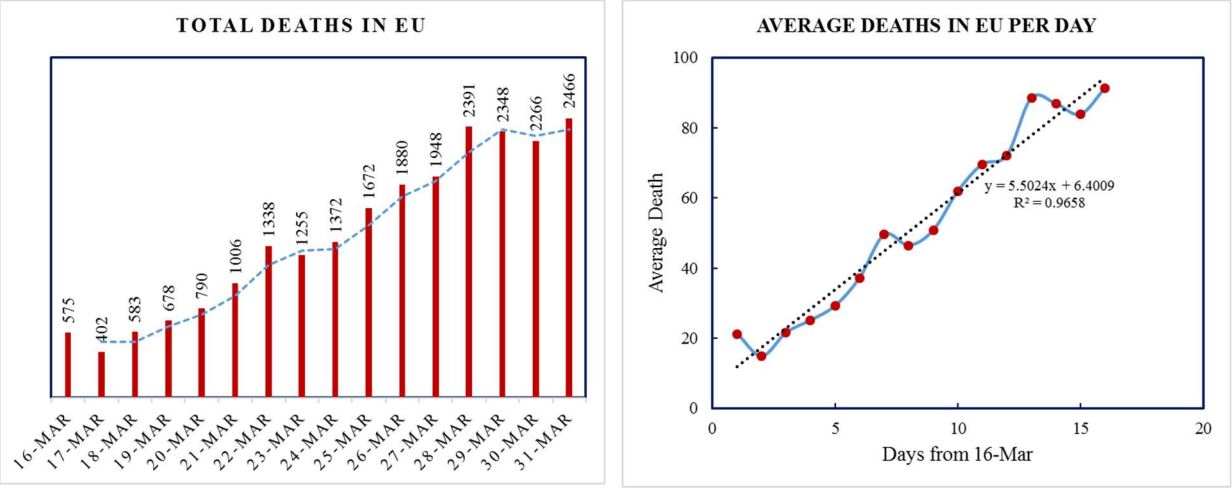
The Coronaviruses under the microscope have a crown-like appearance that is why this family of viruses are called corona which is a Latin word means crown. Coronavirus family are common in human and many different animals, including camels, cattle, cats, and bats. In human Coronavirus spreads mainly from person to person, mostly through breathing droplets produced when an infected person talks directly to another person or when he/she coughs or sneezes. Generally Coronavirus is spreading more efficiently than influenza, but less efficient than measles. Coronavirus family which is known as CoV is not new. CoVs have been discovered since the 1960's [1]. For example Zoonotic coronaviruses have discovered and caused human outbreaks, such as the Severe Acute Respiratory Syndrome (SARS) in 2003 and the Middle East Respiratory Syndrome (MERS) since 2012. So far, about 7 coronaviruses have been discovered: Beta coronavirus HCoV-OC43, HCoV-HKU1, Alpha coronavirus HCoV-229E, Alpha coronavirus HCoV-NL63, SARS-CoV, SARS-CoV2 and MERS-CoV [1, 2]. The novel coronavirus known as COVID-19 was emerged to cases in Wuhan, China on 31 December 2019 is a novel type of SARS-CoV2 and genetically groups within Beta coronavirus subgenus Sarbecovirus [3]. One of the main characteristics of COVID-19 is its high reproduction number (R0) defined as mean number of infections produced by a case of an infection in a completely vulnerable population. On 12 January 2020, Chinese government announced the sequence of a novel coronavirus as type of SARS-CoV-2 isolated from some clustered cases. In European countries on 24 January 2020 first case was confirmed in France, 27 January in Germany, and 31 January in Spain, Italy and United Kingdom. As of 21 February 2020 about 47 confirmed cases of COVID-19 in 9 European countries reported: 21 were linked to two clusters in Germany and France, 14 were infected in China. Median case age was 42 years; 25 were male. As at 5 March, in European countries there were 4,250 confirmed cases. As of 17 March, all Europe countries reported confirmed cases of COVID-19, where Montenegro was the last country to report at least one case. As of 18 March, more than 250 million people were in lockdown in Europe [6, 7]. Review of 12 statistical and clinical modeling in Europe shows that the reproductive number for COVID-19 (R0) is at about 3.28, with a median of 2.79. For instance R0 in Italy estimates between 2 and 3. And form 11 European countries R0 estimated to be around of 3.87 on average. However when social distancing or other restriction are in place the population cannot be considered completely vulnerable therefore the effective reproductive number (Re) is defined. In European Union the Re in Germany is one of the lowest and remains around one or below since the 22 March (95% CI $\approx$ 3.0 - 4.7) An obvious decrease in Re happened following the social distancing and restriction rules are placed in several European countries. As of 8 May, in Europe Spain with 221,447 confirmed cases is the highest and Liechtenstein with 83 confirmed cases is the lowest infected countries. As of 19 March, Public Health England, no longer classified COVID-19 as a "High consequence infectious disease". Also, on May 6 German chancellor Angela Merkel announced that the goal of slowing down the virus has been achieved and that the first phase of the pandemic is over in Germany [1, 2, 4, 5, 8].

References: Full list of references are provide at the end of the report.

Data

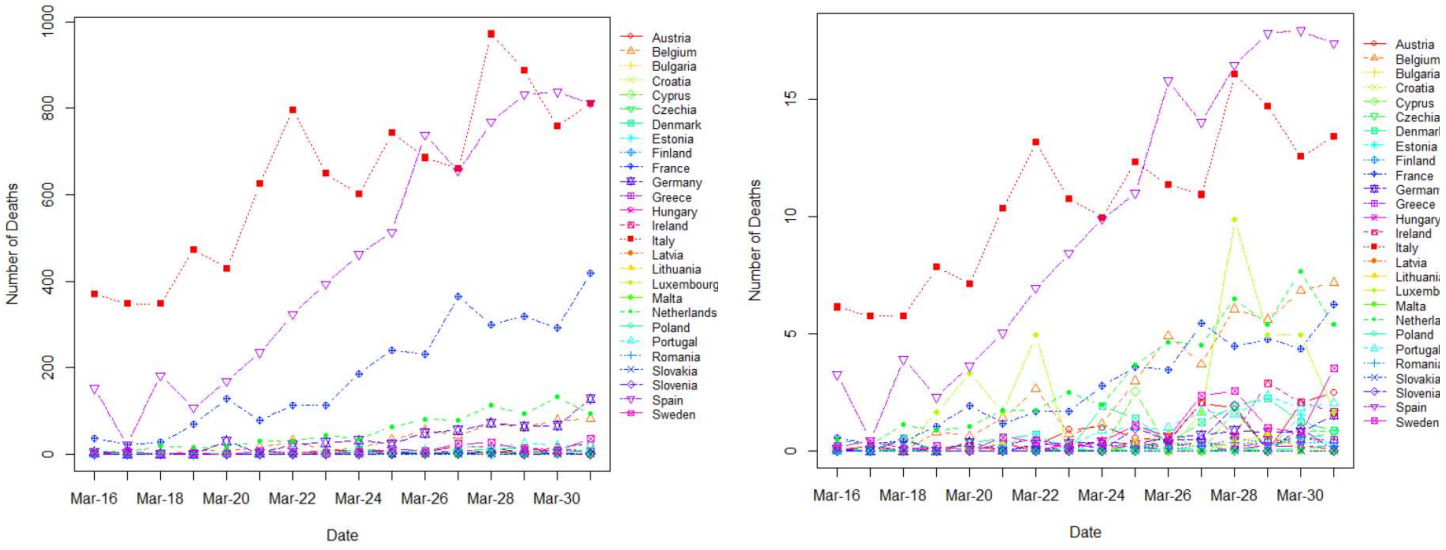
The dataset contains daily deaths related to COVID-19 in the 27 European Union countries from March 16 to March 31 (16 days). In the dataset "Country" column shows the name of the country, "PopulationM" column is the population of country in million people and columns named 16-Mar to 31-Mar show the number of deaths related to COVID-19 by day in each country. Some initial investigation of the data are provided in Figure 1. As it can be seen in figure1 (a) total number of death in the EU increases linearly in the time period of 18 to 28 of March and between 28 and 31 of March the total cases of death in EU per day was almost a plateau. However, average number of death per countries in EU increases linearly in our time period (see Figure 1(b)). Figure 2 shows number of deaths versus day and number of deaths per million inhabitants versus day. According to figure 2:

- Which country has the most deaths over this period?
  Italy had the most death per day which was on 28 of March with 971 cases. During this time period Italy also had the most total number of death among the 27 EU countries.
- Which country has the highest death rate per capita over this period?
  Spain had the most death per capita which was 17.93 death per million inhabitants of the country.
- Which countries have no deaths over this period?
  Latvia, Malta and Slovakia didn't have death cases in this time period.



(a) (b)

Figure 1: Simple review of the data in EU: (a) Total death per day. (b) Average death per day.



(a) (b)

Figure 2: (a) Number of deaths vs. day. (b) Number of deaths per million inhabitants vs. day.

First model

(a) Describe the model in `firstmodel.bug`. Make sure that the following questions are answered by your description:

- What type of (generalized) linear model is this? What does the response variable represent?
- Is it a rate model? What would be the rate (exposure) variable?
- What are the parameters and hyperparameters?
- For the linear portion of the model, does the intercept vary by country? Does the slope (multiplying the centered day variable) vary by country?

(b) List the JAGS code in `firstmodel.bug`.

(c) Summarize the details of your computation, including number of chains, length of burn-in, number of iterations used per chain, any thinning (if used), and effective sample sizes of all parameters. You should use plots to check convergence, but do *not* include them in your report.

Note: Use overdispersed starting values, but make them less extreme if you encounter convergence problems.

(d) Approximate the posterior mean, posterior standard deviation, and 95% central posterior interval for each top-level (hyper)parameter.

(e) Which country has the *highest* posterior *median* value for its intercept? Which country has the *lowest* posterior *median* value for its intercept? (Remark: These are the EU countries with the highest and lowest median per capita death rates for the second half of March, according to this model.)

(f) Approximate the value of (Plummer's) DIC and the associated effective number of parameters. Compare the effective number of parameters with the actual (total) number of parameters.

(a)

- The model is a log-linear regression model (Poisson model). Response variable, deaths [i, j]), is number of COVID-19 deaths recorded in country i (i = 1...27) on day j (j = 1…16).
- As we consider the number of death cases in each day and we considered the population of the countries in the model we can consider the model as a rate model where population of the country is the rate variable.
- Hyperparameters are mu.intercept, sigma.intercept, and slope.
  Parameters are lambda and intercept.
- The slope doesn't vary by country.
  Intercepts for countries are independent and identically distributed so yes we can say it varies by country.

(b)

```
model {

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

    for (j in 1:length(daycent)) {
      deaths[i,j] ~ dpois(lambda[i,j])
      log(lambda[i,j]) <- logpopulation[i] + intercept[i] +
slope*daycent[j]
    }

    intercept[i] ~ dnorm(mu.intercept, 1/sigma.intercept^2)

  }

  slope ~ dnorm(0, 1/100^2)

  mu.intercept ~ dnorm(0, 1/100^2)
  sigma.intercept ~ dunif(0, 100)

}
```

(c)

```
d1 <- list(deaths = data[,3:18],
           logpopulation = log(data$PopulationM),
           daycent = seq(-7.5, 7.5,1))

inits1 <- list(list(slope = 10, mu.intercept = 10, sigma.intercept = 100),
               list(slope = -10, mu.intercept = -10, sigma.intercept = 0.01),
               list(slope = 10, mu.intercept = 10, sigma.intercept = 0.01),
               list(slope = -10, mu.intercept = -10, sigma.intercept = 100))
```

Number of chains = 4
Length of adaptation = 10,000
Length of burn-in = 50,000
Number of iterations used per chain = 200,000
Thinning = 100
Effective sample sizes =

```
> effectiveSize(x1[,c("slope", "mu.intercept","sigma.intercept")])
          slope    mu.intercept sigma.intercept
       8000.000        7661.593        8334.909
```

(d)

	posterior mean	posterior standard deviation	95% central posterior interval
slope	0.1086	0.0015	(0.1055, 0.1115)
mu.intercept	-1.2391	0.3965	(-2.0267, -0.4692)
sigma.intercept	1.9711	0.3414	(1.4208, 2.7535)

(e)

Hungary has the highest posterior median value for its intercept and Portugal has the lowest posterior median value for its intercept.

(f)

```
> dic.samples(m1, 100000)
  |**************************************************| 100%
Mean deviance:  3715
penalty 26.89
Penalized deviance: 3742
```

Effective number of parameters is about 27 which is less than the actual (total) number of parameters which is 29.

Second model

(a)

```
model {

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

    for (j in 1:length(daycent)) {
      deaths[i,j] ~ dpois(lambda[i,j])
      log(lambda[i,j]) <- logpopulation[i] + intercept[i] +
slope[i]*daycent[j]
    }

    intercept[i] ~ dnorm(mu.intercept, 1/sigma.intercept^2)
    slope[i] ~ dnorm(mu.slope, 1/sigma.slope^2)
  }


  mu.intercept ~ dnorm(0, 1/100^2)
  sigma.intercept ~ dunif(0, 100)

  mu.slope ~ dnorm(0, 1/100^2)
  sigma.slope ~ dunif(0, 100)
}
```

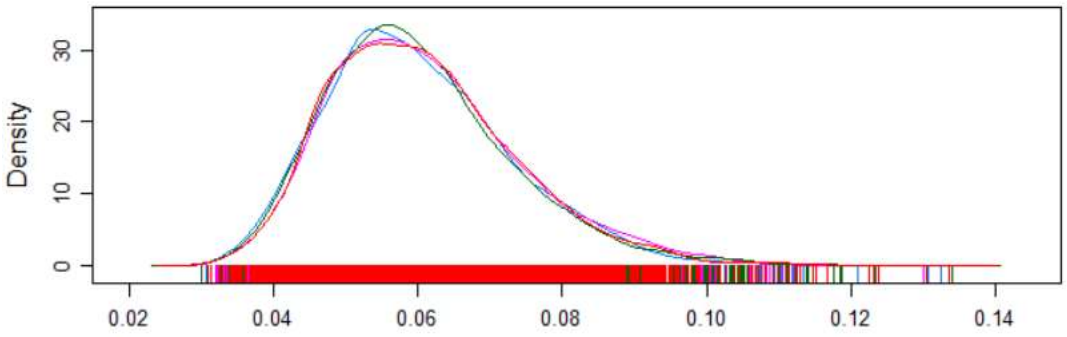
```
d1 <- list(deaths = data[,3:18],
           logpopulation = log(data$PopulationM),
           daycent = seq(-7.5, 7.5,1))

inits1 <- list(list(mu.intercept = 100, sigma.intercept = 100, mu.slope = 10, sigma.slope = 100),
               list(mu.intercept = -100, sigma.intercept = 0.01, mu.slope = -10, sigma.slope = 0.01),
               list(mu.intercept = 100, sigma.intercept = 0.01, mu.slope = 10, sigma.slope = 0.01),
               list(mu.intercept = -100, sigma.intercept = 100, mu.slope= -10, sigma.slope = 100))
```

(b)

Number of chains = 4
Length of adaptation = 10,000
Length of burn-in = 50,000
Number of iterations used per chain = 300,000
Thinning = 100
Effective sample sizes =

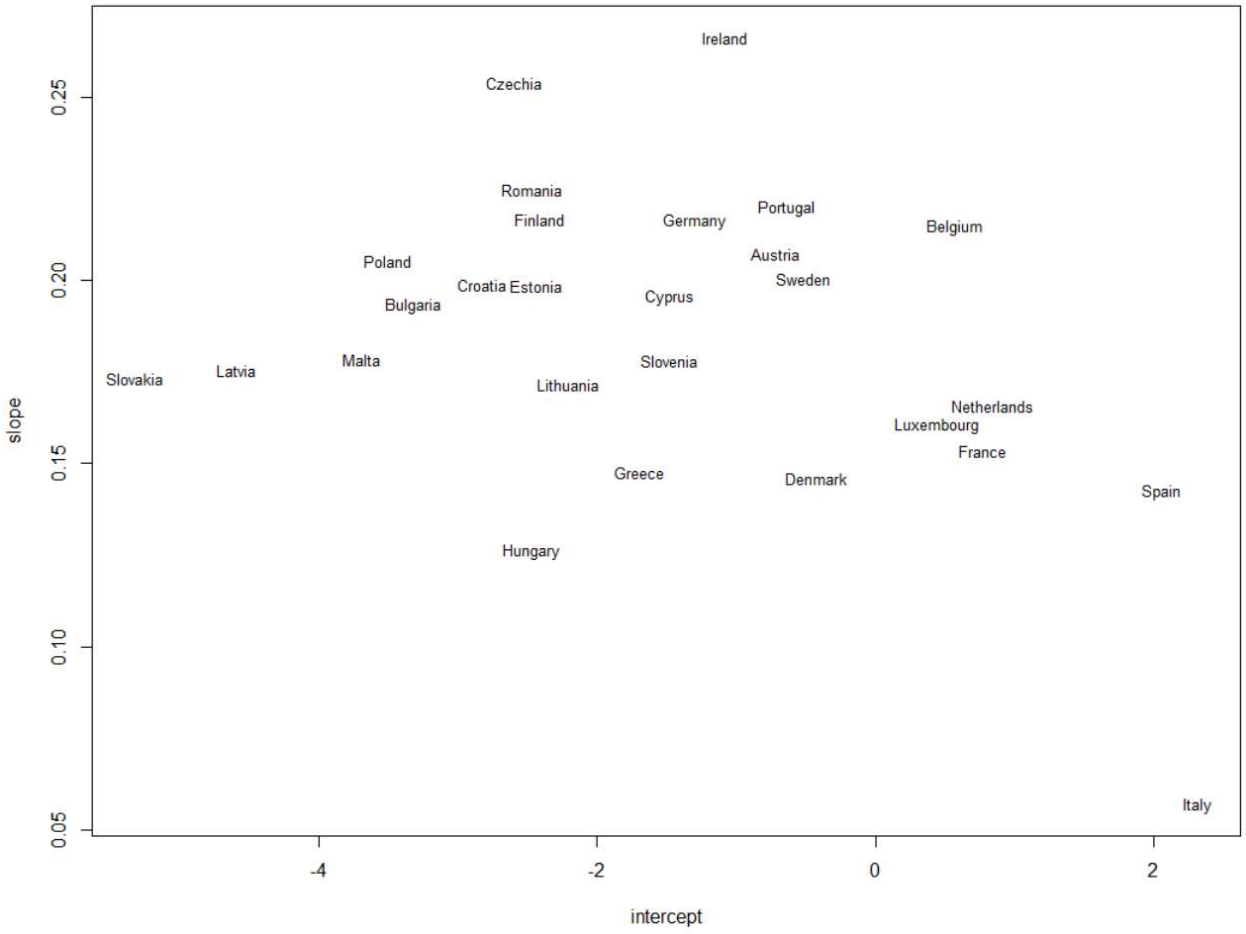
```
> effectiveSize(x1[,c("mu.slope","sigma.slope", "mu.intercept","sigma.intercept")])
       mu.slope     sigma.slope    mu.intercept sigma.intercept
       11710.69        11853.25        11449.80        12000.00
```

(c)



As pick of graph of approximate posterior density of σ_{slope} is near zero shows that the variation between slopes for different countries was low it does not suggest.

(d)



(e)

```
> dic.samples(m1, 100000)
  |**************************************************| 100%
Mean deviance:  2590
penalty 41.71
Penalized deviance: 2631
```

Effective number of parameters is about 42. As DIC is lower for the second model yes it is a better model than the first model. Second model is preferred.

Conclusions

In this project data from of daily deaths related to COVID-19 in the 27 European Union (EU) member states for the second half of March 2020 investigated. First the data briefly investigated and then two Bayesian model was used to further explorer the data and try to fit a Bayesian model. From the data I found the following:

- Total number of death in the EU increases linearly in this time period.
- Italy had the most death per day.
- Spain had the most death per capita.
- Latvia, Malta and Slovakia didn't have death cases in this time period.

Two Bayesian model fitted using JAGS in R. Both model are a log-linear regression model (Poisson model). For the first model similar slope for different countries is assumed. For the second model, however, slope for different countries assumed to be independent and identically distributed.

It is found that the second model is better than the first model.

Main summary of the models are as follows:

First model:

Number of chains = 4
Length of adaptation = 10,000
Length of burn-in = 50,000
Number of iterations used per chain = 200,000
Thinning = 100
DIC: 3715
Effective number of parameters = 27

Second model:

Number of chains = 4
Length of adaptation = 10,000
Length of burn-in = 50,000
Number of iterations used per chain = 300,000
Thinning = 100
DIC: 2590
Effective number of parameters = 42

References

[1] Disease background of COVID-19. European Centre for Disease Prevention and Control. Website: https://www.ecdc.europa.eu/en/2019-ncov-background-disease.
[2] Yin, Y., and Wunderink, R., G. (2018). "MERS, SARS and other coronaviruses as causes of pneumonia. Respirology (Carlton, Vic)". 23, 2, 130-137.
[3] World Health Organization (WHO). WHO Statement regarding cluster of cases in Wuhan, China 2020. Website: https://www.who.int/china/news/detail/09-01-2020-who-statementregarding-cluster-of-pneumonia-cases-in-wuhan-china.
[4] Riccardo, F., Ajelli, M., Andrianou, X., Bella, A., Del Manso, M., Fabiani M, et al. "Epidemiological characteristics of COVID-19 cases in Italy and estimates of the reproductive numbers one month into the epidemic". medRxiv. 2020:2020.04.08.2005686.
[5] Flaxman, S., Mishra, S., Gandy, A., Unwin, H., Coupland, H., Mellan, T. et al. (2020). "Estimating the number of infections and the impact of non-pharmaceutical interventions on COVID-19 in 11 European countries" Imperial Colleage London. Website: https://spiral.imperial.ac.uk:8443/handle/10044/1/77731.

[6] First cases of coronavirus disease 2019 (COVID-19) in the WHO European Region, 24 January to 21 February 2020. Website: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7068164/.
[7] COVID-19 pandemic in Europe. Wikipedia. Website: https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Europe.
[8] COVID-19 pandemic in Germany. Wikipedia. Website: https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Germany

Appendix

```
#////////////////////////////
#Read data
#////////////////////////////
data = read.csv(file.choose(),sep=",",header = TRUE) # read EUCOVIDdeaths.csv
nn <- ncol(data) - 2
#////////////////////////////
#First Graph
#////////////////////////////
matplot(t(data[,-c(1,2)]), type = "b", xlab = "Date", ylab = "Number of Deaths", xaxt='n', pch = c(1:20), lty = c(1:20), col = rainbow(14))

#Create x axis label
a=c()
for (i in 16:31) {
  d=paste("Mar-",i,sep="")
  a=c(a,d)
}
axis(side=1,at=1:nn,labels= a)

#Put the legend outside the graph
library(varhandle)
par(mar =  c(5, 4, 4, 6) + 0.1)
legend(16.8,1000, unfactor(data$Country), pch = c(1:20), lty = c(1:20), col = rainbow(14), cex=0.8, inset=c(+0,+1000), bty = "n", xpd=TRUE)
#////////////////////////////
#Second Graph
#////////////////////////////
#prepare the data based on per capita deaths
perCapita = data
for(i in 1: nrow(perCapita)){
  population = perCapita[i,2]
  for(j in 3: ncol(perCapita)){
    perCapita[i,j] = perCapita[i,j] / population
  }
}



#////////////////////////////
#Second Graph
#////////////////////////////
matplot(t(perCapita[,-c(1,2)]), type = "b", xlab = "Date", ylab = "Number of Deaths", xaxt='n', pch = c(1:20), lty = c(1:20), col = rainbow(14))
axis(side=1,at=1:nn,labels= a)
par(mar =  c(5, 4, 4, 6) + 0.1)
legend(16.8,18, unfactor(data$Country), pch = c(1:20), lty = c(1:20), col = rainbow(14), cex=0.8, inset=c(+0,+1000), bty = "n", xpd=TRUE)
#////////////////////////////
#////////////////////////////
#First Model JAGS
#////////////////////////////
#////////////////////////////
library(rjags)

d1 <- list(deaths = data[,3:18],
           logpopulation = log(data$PopulationM),
           daycent = seq(-7.5, 7.5,1))

inits1 <- list(list(slope = 10, mu.intercept = 100, sigma.intercept = 100),
               list(slope = -10, mu.intercept = -100, sigma.intercept = 0.01),
               list(slope = 10, mu.intercept = 100, sigma.intercept = 0.01),
               list(slope = -10, mu.intercept = -100, sigma.intercept = 100))

m1 <- jags.model(file.choose(), d1, inits1, n.chains=4, n.adapt=10000) #select JAGS file
update(m1, 50000) # burn-in
x1 <- coda.samples(m1, c("slope","mu.intercept","sigma.intercept", "intercept", "lambda"), n.iter=300000, thin=100)

#Trace plot
plot(x1[,c("slope", "mu.intercept","sigma.intercept")])

gelman.diag(x1[,c("slope", "mu.intercept","sigma.intercept")], autoburnin=FALSE)

#effective size of parameters
effectiveSize(x1[,c("slope", "mu.intercept","sigma.intercept")])

summary(x1[,c("slope", "mu.intercept","sigma.intercept")])

# The value of (Plummer's) DIC and the associated effective number
dic.samples(m1, 100000)
```

```
#////////////////////////////
#////////////////////////////
#Second Model JAGS
#////////////////////////////
#////////////////////////////

d1 <- list(deaths = data[,3:18],
           logpopulation = log(data$PopulationM),
           daycent = seq(-7.5, 7.5,1))

inits1 <- list(list(mu.intercept = 100, sigma.intercept = 100, mu.slope = 10, sigma.slope = 100),
               list(mu.intercept = -100, sigma.intercept = 0.01, mu.slope = -10, sigma.slope = 0.01),
               list(mu.intercept = 100, sigma.intercept = 0.01, mu.slope = 10, sigma.slope = 0.01),
               list(mu.intercept = -100, sigma.intercept = 100, mu.slope= -10, sigma.slope = 100))

m1 <- jags.model(file.choose(), d1, inits1, n.chains=4, n.adapt=10000) #select second JAGS file
update(m1, 50000) # burn-in
x1 <- coda.samples(m1, c("mu.intercept","sigma.intercept","mu.slope","sigma.slope", "intercept", "slope"), n.iter=300000, thin=100)

#Trace plot
plot(x1[,c("mu.slope","sigma.slope", "mu.intercept","sigma.intercept")])

gelman.diag(x1[,c("mu.slope","sigma.slope", "mu.intercept","sigma.intercept")], autoburnin=FALSE)

#effective size of parameters
effectiveSize(x1[,c("mu.slope","", "mu.intercept","sigma.intercept")])

summary(x1)

#Density plot of sigma_slope
require(lattice)
densityplot(x1[, c("sigma.slope")])


#posterior expected intercept and slope
intercept = ((summary(x1))$statistics)[paste("intercept[",1:27,"]", sep=""),"Mean"]
slope = ((summary(x1))$statistics)[paste("slope[",1:27,"]", sep=""),"Mean"]

plot(intercept, slope, type="n")
text(intercept, slope, data$Country, cex=0.8)

# The value of (Plummer's) DIC and the associated effective number
dic.samples(m1, 100000)
```