amana4@illinois.edu

Aman Arora

STAT 578

05/05/2020

COVID-19 in EU – Modelling using JAGS and R

1. Introduction

Covid-19 is infectious disease caused by a novel corona virus (SARS-COV2). The virus is known to have originated in Hubei province in China in Dec 2019. Since then COVID-19 has spread across the world and has become a worldwide pandemic. With respect to EU the index case was first found in Jan 2020 in France [1] and by March had spread across almost all countries in EU. Countries in EU were impacted differently due to their diverse risk profiles for COVID19 due to following major factors:

- **Exposure to international travelers -** with large and connected countries like Italy/France/Spain receiving high number of infectious travelers from China vs smaller countries like Finland/Slovakia .
- **Demographics** – counties with older populations and high population densities (like Italy) at higher risk of death and infection respectively.
- **Response –** Measures taken to respond to pandemic e.g. severity of lockdown and travel restrictions varied across governments .

All these factors resulted in countries getting themselves placed at different points in the expected death rate curves. For example by mid March, 2020 Italy and Spain had high number of deaths already and were trending rapidly towards the peak while other large countries like Germany and France were at relatively lower point in the curve. By end of March, 2020 Italy and Spain seemed to have reached the peak deaths and were trending downwards while Germany and France were still trending upwards towards their respective peaks. There were also exceptions like Malta and Slovakia which had no reported deaths from in March but only started reporting deaths from April, 2020 onwards [2].

2. Data

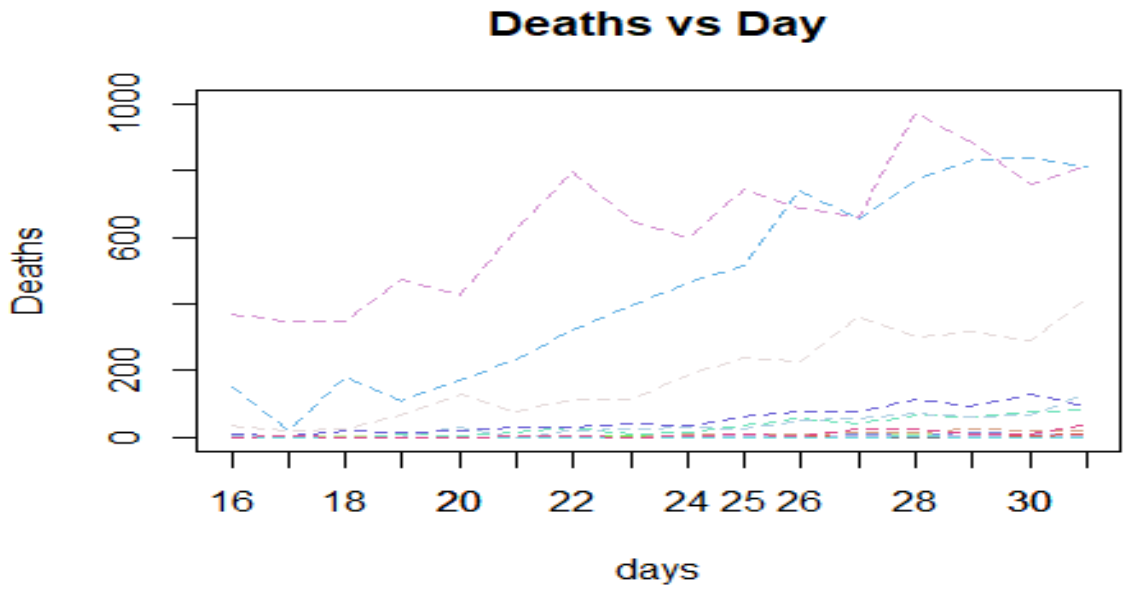
The data file have following variables:

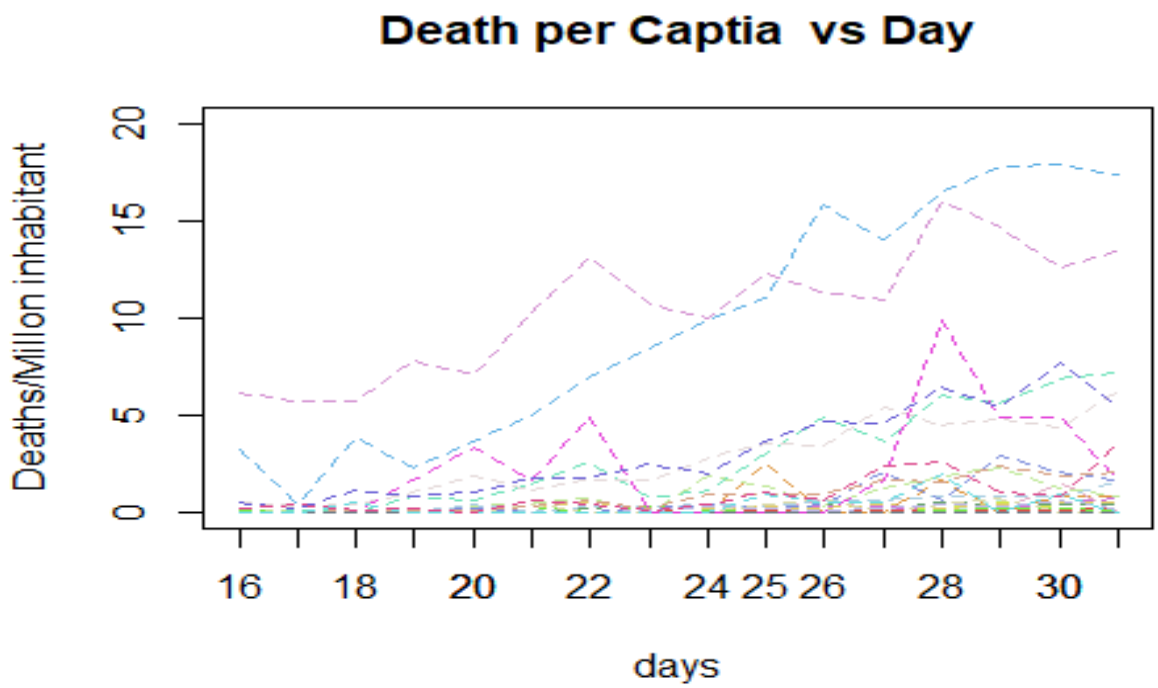
- Country Name
- Population of the country in millions
- Daily deaths reported from March 16th to March 31st for a country.

[1] Wikipedia, The Free Encyclopedia. Retrieved 5th May, 2020 from https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Europe

[2] ECDC website, Retrieved 5th May, 2020 from https://www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases

amana4@illinois.edu





As we can see from the plots - this time period captures deaths and death rates for countries with varying deaths per capita over time and a varying starting death count (at day 16). Also this time period seems to have captured increasing deaths per captia for most countries (the escalating part of epidemic curve) with the death rate decreasing only near the end of the time period for few countries.

- Country with most deaths (cumulative) over the period is **Italy.**
- Country with highest death rate per capita (cumulative) over the period is **Italy** as well.
- Countries with no deaths are: **Latvia, Malta, Slovakia.**

3. First Model

3.1 Description of the model

The first model that will be used to model the number of deaths is a relatively simple **Poisson log linear regression model**. The response variable represents the count of deaths for country i on day j. It's a rate model wherein the exposure variable is the population of the country in millions.

The hyperparameters are:

- mu.intercept
- sigma.intercept
- slope.

The parameters are:

- intercept[i] where i ranges from 1 to 27 (country).

The intercept varies by country as we can see above each country has its own intercept. The slope is common and identical for all the countries in this model.

3.2 JAGS code

```
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)
```

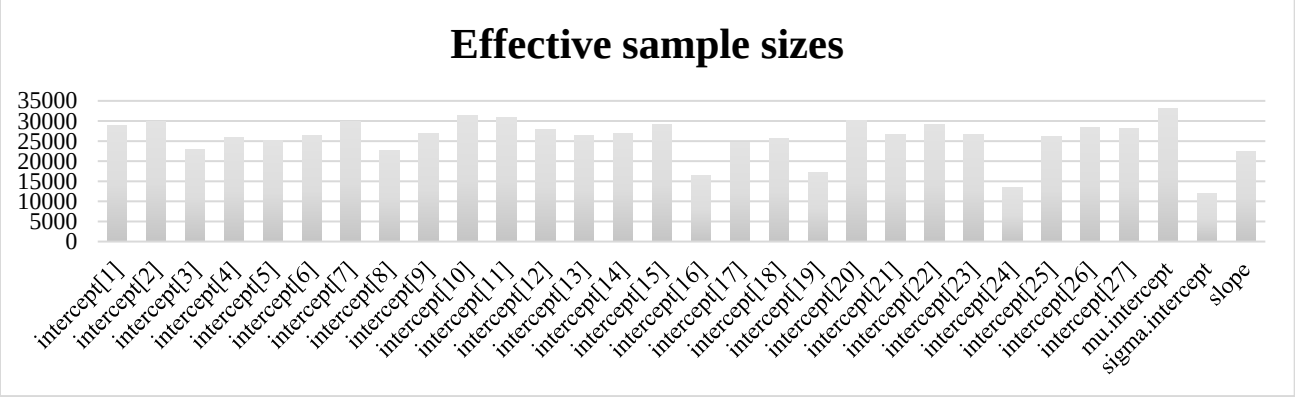
```
  mu.intercept ~ dnorm(0, 1/100^2)

  sigma.intercept ~ dunif(0, 100)

}
```

3.2 Details of the computation

The model was run using R (3.6.3)/JAGS(4.3.0) with over dispersed starting values for top level parameters with following configuration:

- 4 MCMC (Monte Carlo Markov chains).
- 10000 iteration each.
- Burn in length of 1000
- 1000 iterations used for adaptation
- No thinning. Random seed was set for reproducibility

Effective samples sizes are in the chart below:



3.3 Posterior Statistics

Mean and SD

Variable	Country	Mean	SD
intercept[1]	Austria	-0.409748242	0.096950855
intercept[2]	Belgium	0.900045181	0.044132265
intercept[3]	Bulgaria	-3.044175658	0.396645159
intercept[4]	Croatia	-2.526695221	0.394073201
intercept[5]	Cyprus	-1.192299462	0.38255511
intercept[6]	Czechia	-2.090312725	0.203080026
intercept[7]	Denmark	-0.318492351	0.114384786
intercept[8]	Estonia	-2.14767956	0.573023405
intercept[9]	Finland	-2.055205157	0.275222694
intercept[10]	France	0.883971533	0.018878874
intercept[11]	Germany	-0.959396878	0.04166521
intercept[12]	Greece	-1.589220285	0.156894883
intercept[13]	Hungary	-2.411748292	0.247443754

intercept[14]	Ireland	-0.537581414	0.139126676
intercept[15]	Italy	2.228595959	0.010457027
intercept[16]	Latvia	-4.277435424	1.176656659
intercept[17]	Lithuania	-2.009878175	0.377505347
intercept[18]	Luxembourg	0.599518234	0.223790081
intercept[19]	Malta	-3.369452936	1.297994546
intercept[20]	Netherlands	1.004665148	0.03441014
intercept[21]	Poland	-3.197147992	0.187764719
intercept[22]	Portugal	-0.288685157	0.083941863
intercept[23]	Romania	-2.08486274	0.150542446
intercept[24]	Slovakia	-5.028951784	1.097456943
intercept[25]	Slovenia	-1.266880113	0.302318952
intercept[26]	Spain	2.143027912	0.012261881
intercept[27]	Sweden	-0.244427601	0.082928707
mu.intercept	-	-1.231716014	0.398448336
sigma.intercept	-	1.971525794	0.333952574
slope	-	0.108609771	0.001543993

95% posterior intervals

Variable	Country	2.50%	25%	50%	75%	97.50%
intercept[1]	Austria	-0.60585	-0.47369	-0.407799047	-0.343873603	-0.224479499
intercept[2]	Belgium	0.812436	0.870418	0.900385241	0.929825217	0.985924055
intercept[3]	Bulgaria	-3.89338	-3.29395	-3.01957898	-2.768209323	-2.336866494
intercept[4]	Croatia	-3.3559	-2.781	-2.505245389	-2.251475106	-1.812639792
intercept[5]	Cyprus	-2.00609	-1.43202	-1.170296264	-0.925322989	-0.509498107
intercept[6]	Czechia	-2.50848	-2.22233	-2.083236358	-1.949842978	-1.714420385
intercept[7]	Denmark	-0.54796	-0.39446	-0.316894912	-0.240943735	-0.102241592
intercept[8]	Estonia	-3.41211	-2.49361	-2.097852413	-1.743096109	-1.169649988
intercept[9]	Finland	-2.62659	-2.23396	-2.042041328	-1.864057805	-1.549208666
intercept[10]	France	0.846838	0.871179	0.884073506	0.896738994	0.920633668
intercept[11]	Germany	-1.0421	-0.98761	-0.958982998	-0.931041961	-0.879252252
intercept[12]	Greece	-1.90833	-1.69203	-1.585834747	-1.481863685	-1.291744262
intercept[13]	Hungary	-2.91858	-2.57335	-2.401875402	-2.239431061	-1.954620999
intercept[14]	Ireland	-0.81829	-0.62868	-0.536169343	-0.442319272	-0.273216386
intercept[15]	Italy	2.207964	2.221616	2.228623204	2.235657834	2.249072989
intercept[16]	Latvia	-7.0262	-4.94705	-4.118000177	-3.434735072	-2.443651993
intercept[17]	Lithuania	-2.80386	-2.24908	-1.989379331	-1.747346164	-1.332116036
intercept[18]	Luxembourg	0.129308	0.455389	0.609197182	0.754361133	1.012655486
intercept[19]	Malta	-6.35308	-4.11334	-3.210821941	-2.4452021	-1.31657945
intercept[20]	Netherlands	0.936311	0.981504	1.004935952	1.02786205	1.071739889
intercept[21]	Poland	-3.5837	-3.31912	-3.19034455	-3.068321756	-2.845427418
intercept[22]	Portugal	-0.45791	-0.34405	-0.287099493	-0.231907881	-0.126174392

intercept[23]	Romania	-2.39103	-2.18492	-2.080672044	-1.980478966	-1.800242137
intercept[24]	Slovakia	-7.61332	-5.63298	-4.869698063	-4.259949184	-3.325063014
intercept[25]	Slovenia	-1.90015	-1.46095	-1.251672462	-1.056815538	-0.711403277
intercept[26]	Spain	2.118909	2.134763	2.143016453	2.151308226	2.166990783
intercept[27]	Sweden	-0.41074	-0.2991	-0.243051856	-0.188589394	-0.0845442
mu.intercept	-	-2.03922	-1.48666	-1.224072381	-0.966963685	-0.462760853
sigma.intercept	-	1.439294	1.734159	1.928118718	2.163714173	2.749293675
slope	-	0.10558	0.107555	0.108619203	0.109664204	0.111627423

As we can see from the table above the country with the highest posterior median value (50%) of intercept is **Italy** and lowest median value is **Slovakia**.

Model evaluation:

The DIC Mean deviance is: 3715
Penalty is 26.91
Final Penalized deviance: 3742

The actual no of parameters are 30 (27 intercepts, 1 slope, 1 mu.intercept and 1 sigma.intercept)
With hierarchical modelling the no of effective parameters thus got reduced to 26.91.

4. Second Model

From visual inspection of the plots it would seem the countries may have their distinct slope for no. of deaths over days. The second model would use this observation to model the slopes for each country independently (conditionally given common mean and variance) from a normal distribution unlike first model as below:

JAGS code

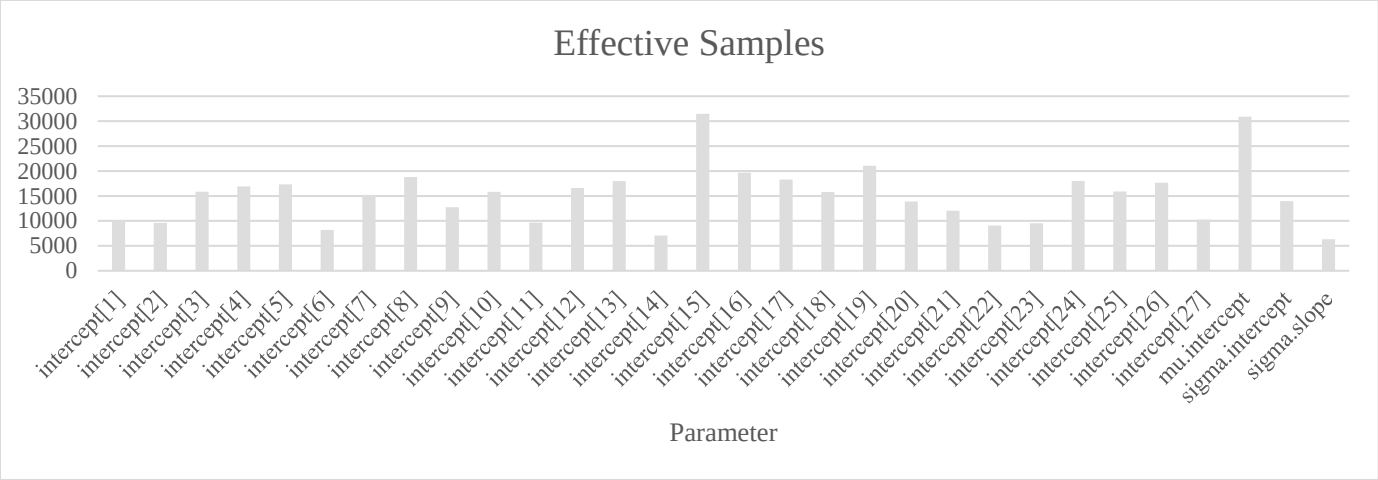
```
model {
 for (i in 1:length(logpopulation)) {
  for (j in 1:length(daycent)) {
   deaths[i,j] ~ dpois(lambda[i,j])
   log(lambda[i,j]) <- logpopulation[i] + intercept[i] + slope[i]*daycent[j]
  }
  intercept[i] ~ dnorm(mu.intercept, 1/sigma.intercept^2)
  slope[i] ~ dnorm(mu.slope, 1/sigma.slope^2)
 }
 mu.intercept ~ dnorm(0, 1/100^2)
 sigma.intercept ~ dunif(0, 100)
 mu.slope ~ dnorm(0, 1/100^2)
 sigma.slope ~ dunif(0, 100)
```

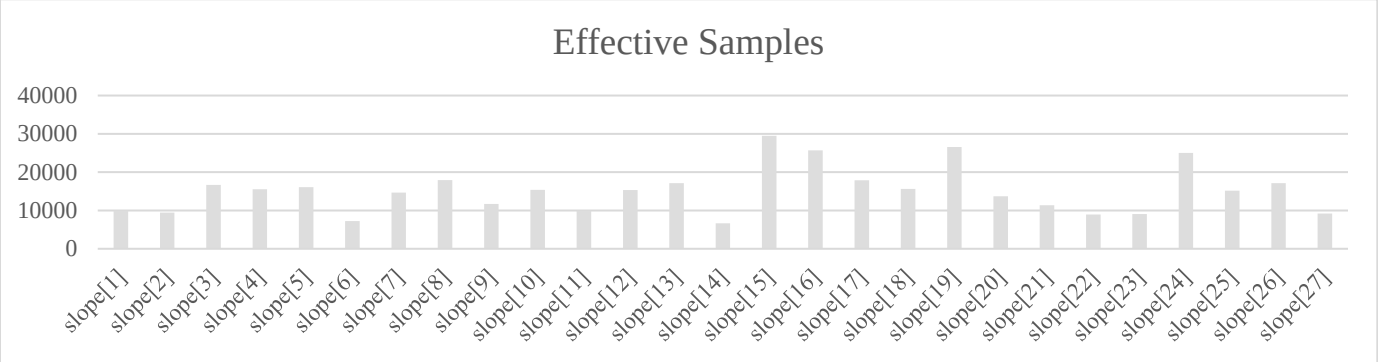
}

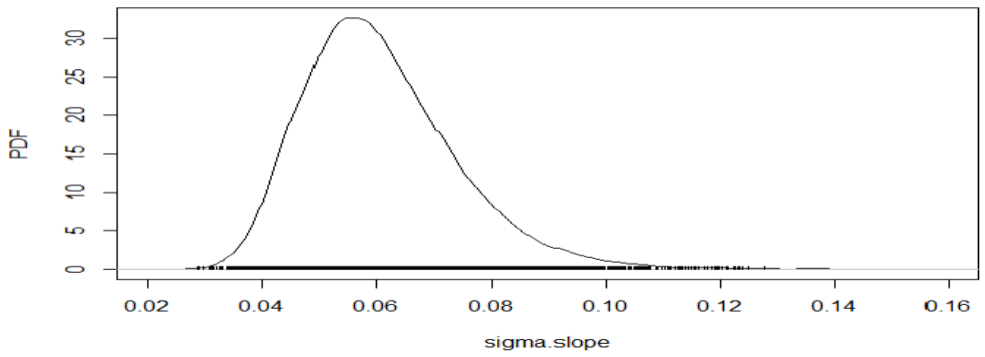
The model was run using R (3.6.3)/JAGS(4.3.0) with over dispersed starting values for top level parameters with following configuration:

- 4 MCMC (Monte Carlo Markov chains).
- 10000 iteration each.
- Burn in length of 1000, 1000 iterations used for adaptation
- No thinning. Random seed was set for reproducibility

Effective samples sizes are in the chart below:

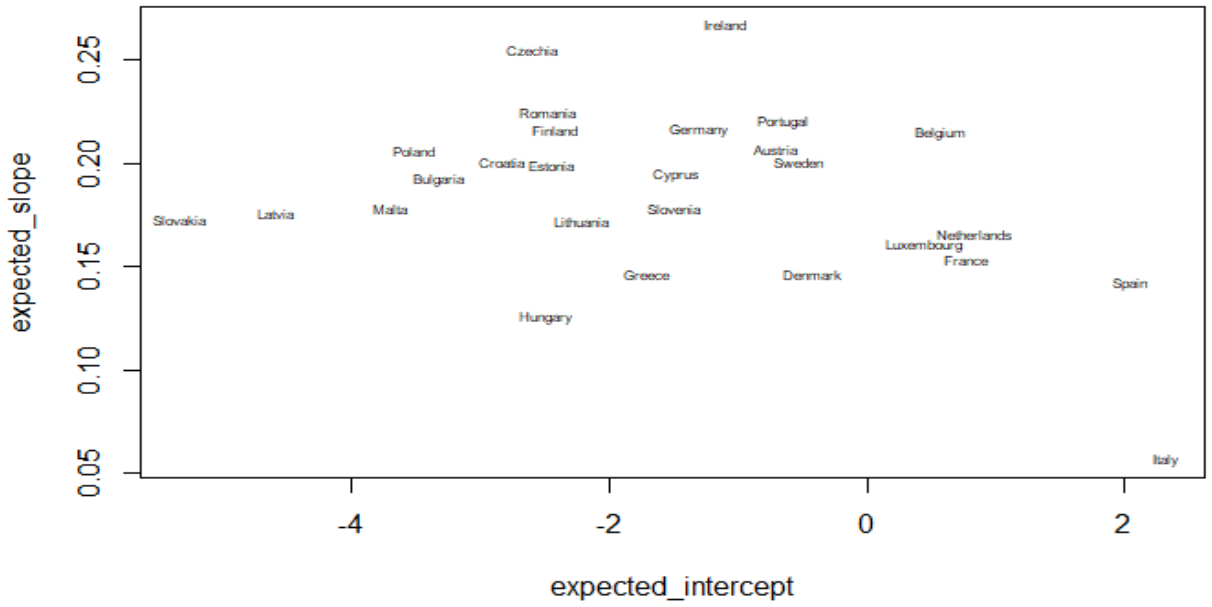






From the density plot (above) of the standard deviation of the slope we can clearly see there is a variation in slope of the countries and second model should thus likely give a better fit than the first model (with assumption of identical slope).

The plot below is for expected slope vs intercept of the each country



Model evaluation:

```
Mean deviance:  2590
penalty 41.74
Penalized deviance: 2632
```

Based on overall DIC score 2nd model is better.

5 Conclusions

In the first model we are modelling the log(no of deaths) as Poisson process with a rate parameter that is linear function of time with an intercept that's country specific but slope is shared. The 2nd model adds additional complexity by allowing slope to vary by country as well.

As we can see from expected slope vs expected intercept graph (above) variation in slope among countries is significant. This is also evident from the non zero variance of the slope parameter in 2nd model.

The first model allowed country specific intercept but identical slope resulting in model deviance being high. The second model does not assume identical slope and thus provides a better fit indicated by lower mean deviance, the penalty for 2nd model is higher since it adds additional parameters (complexity) to model slope individually but overall the 2nd model gives lower DIC score due to much better fit to data.

Appendix

Section 2 - R code:

```
# Load and format the data
library(randomcoloR)
df = read.csv("C://Users//amana//Downloads//EUCOVIDdeaths.csv",header=TRUE)
num_days <- ncol(df)
num_countries <- nrow(df)
palette <- distinctColorPalette(num_countries)
days <- 16:31

# Plot deaths vs days
plot(days,(df[1,3:num_days]),ylim = c(0, 1000),col = palette[0],ylab = "Deaths",lty =2)
axis(1,at=c(16:31))
for (i in 2:num_countries)
   lines(days,(df[i,3:num_days]),ylim = c(0, 1000),col = palette[i],lty =2)

# Plot death rate vs days
plot(days,(df[1,3:num_days])/df[1,2],ylim = c(0, 20),col = palette[0],ylab = "Deaths/Millon inhabitant",lty =2)
axis(1,at=c(16:31))
for (i in 2:num_countries)
   lines(days,(df[i,3:num_days])/df[i,2],ylim = c(0, 20),col = palette[i],lty =2)

#Country with max deaths
death_sum <- rowSums(df[3:num_days])
max_index <- which.max(death_sum)
df[max_index,1]
death_sum <- rowSums(df[3:num_days])
max_index <- which.max(death_sum)
df[max_index,1]

#Country with max deaths per capita
rate_sum <- vector(,num_countries)
for (i in 1:num_countries)
  rate_sum[i] <- death_sum[i]/df[i,2]
```

```
max_index <- which.max(rate_sum)
df[max_index,1]
#Country with no deaths
df[which(death_sum==0),1]
```

Section 3

```
# set over dispersed starting points and get data frame ready for JAGS
d3 <- list(deaths = df[,-1:-2],logpopulation  = log(df$PopulationM), daycent = days - mean(days))
inits2 <-
list(list(slope = 50,mu.intercept = 200,sigma.intercept = 100, .RNG.name="base::Wichmann-Hill",.RNG.seed=81),
list(slope = 50,mu.intercept = -200,sigma.intercept = 100,.RNG.name="base::Wichmann-Hill",.RNG.seed=82),
list(slope = -50,mu.intercept = 200,sigma.intercept = 10^-3,.RNG.name="base::Wichmann-Hill",.RNG.seed=83),
list(slope = -50,mu.intercept = -200,sigma.intercept = 10^-3,.RNG.name="base::Wichmann-Hill",.RNG.seed=84))
library(rjags)
#Run JAGS model
m1 <- jags.model("C://Users//amana//Downloads//firstmodel.bug", d3, inits2, n.chains=4, n.adapt=1000)
update(m1, 1000) # burn-in
#Monitor top level parameters
x1 <- coda.samples(m1, c("mu.intercept","sigma.intercept","slope","intercept"),n.iter=10000)
#Check for convergence
gelman.diag(x1, autoburnin=FALSE)
gelman.plot(x1)
#Find effective sample size
effectiveSize(x1)
#Summarize results
x2<- summary(x1)
x3 <- x2$quantiles[,3]
#find country with max and min intercept
max_index <- which.max(x3)
df[max_index,1]
min_index <- which.min(x3)
df[min_index,1]
#Compute DIC
dic.samples(m1,10000)
```

Section 4

```
#Setup data and starting points to  run the second model
```

```
inits2 <-
list(list(mu.slope = 50,mu.intercept = 200, sigma.slope = 100, sigma.intercept = 100, .RNG.name="base::Wichmann-Hill",.RNG.seed=81),
list(mu.slope = 50,mu.intercept = -200,sigma.slope = 10^-3,sigma.intercept = 100,.RNG.name="base::Wichmann-Hill",.RNG.seed=82),
list(mu.slope = -50,mu.intercept = 200,sigma.slope = 100,sigma.intercept = 10^-3,.RNG.name="base::Wichmann-Hill",.RNG.seed=83),
list(mu.slope = -50,mu.intercept = -200,sigma.slope = 10^-3,sigma.intercept = 10^-3,.RNG.name="base::Wichmann-Hill",.RNG.seed=84))
m1 <- jags.model("C://Users//amana//Downloads//secondmodel.bug", d3, inits2, n.chains=4, n.adapt=1000)
# burn-in
update(m1, 1000)
#Monitor top level parameters
x1 <- coda.samples(m1, c("mu.intercept","sigma.intercept","slope","intercept","sigma.slope"),n.iter=10000)
#check for convergence
gelman.diag(x1, autoburnin=FALSE)
gelman.plot(x1)
#Find effective sample sizes
effectiveSize(x1)
#Density plot for sigma.slope
x3 <-as.matrix(x1)
densplot(x1[,c("sigma.slope")], xlab = "sigma.slope", ylab ="PDF")
#Plot expected intercept vs slope
post_intercepts_samples <- x3[, paste("intercept[",1:nrow(df),"]", sep="")]
expected_intercept <- apply(post_intercepts_samples,2,mean)
post_slope_samples <- x3[, paste("slope[",1:nrow(df),"]", sep="")]
expected_slope <- apply(post_slope_samples,2,mean)
plot(expected_intercept,expected_slope, type ="n")
text(expected_intercept,expected_slope,df[,1],cex =.5)
#Compute DIC
dic.samples(m1,10000)
```