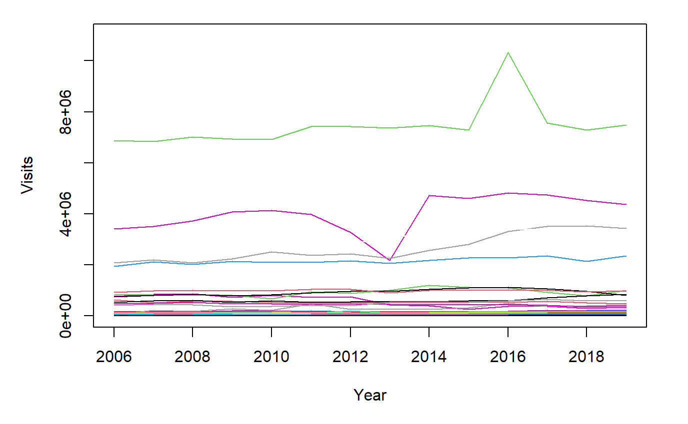
A5

Shyam Shah

10/04/2021

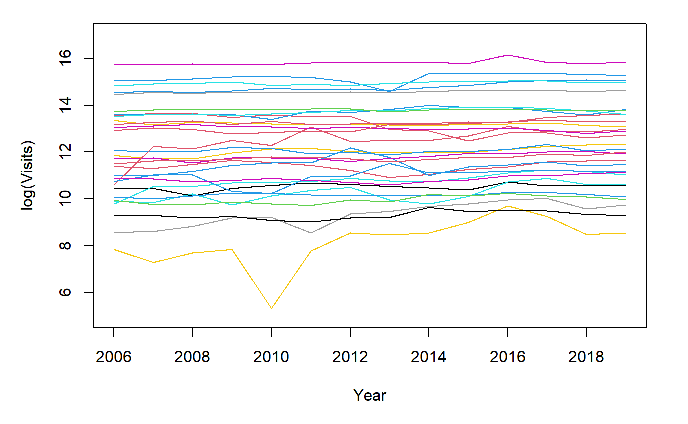
1 (a) (i)

```r
data <- read.csv("./usparkvisits.csv", header=TRUE) #read data
xs <- 2006:2019 #get list of years
plot(xs, data[1,][2:15], col=runif(1, 0, 100), type="l",  ylim=c(0,11000000), xlab="Year", ylab="Visits") #plot the first data point as a line
for (i in 1:length(data[,1])) {
  lines(xs, data[i,][2:15], col=runif(1, 0, 100)) #plots the rest of the data on the same graph
}
```



1 (a) (ii)

```r
plot(xs, log(data[1,][2:15]), col=runif(1, 0, 100), type="l",  ylim=c(5,17), xlab="Year", ylab="log(Visits)") #plot the first data point as a line
for (i in 1:length(data[,1])) {
  lines(xs, log(data[i,][2:15]), col=runif(1, 0, 100)) #plots the rest of the data on the same graph
}
```

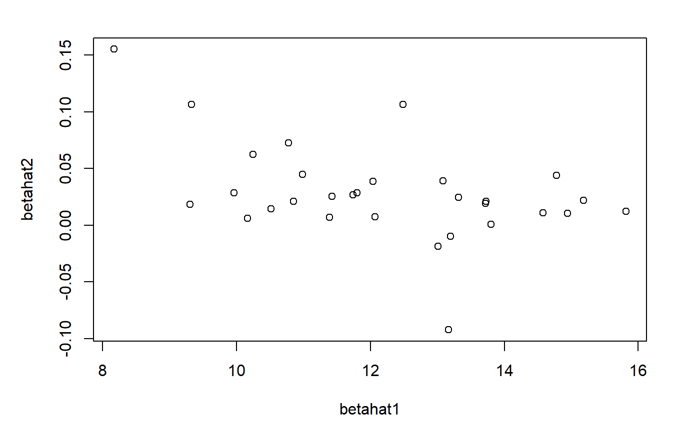


1 (b) (i)

```r
library(MASS)
plaindata <- data[,2:15] #get rid of the noise in the data
xs <- 1:14
xcentered <- xs-mean(xs) #center x values
coeffs <- matrix(NA, length(plaindata[,1]), 2) #create matrix to store values

for (i in 1:length(plaindata[,1])){
 classic.mod <- lm (unlist(log(plaindata[i,])) ~ xcentered) #perform classical linear regression
 coeffs[i,] <- as.matrix(coef(classic.mod)) #store values of the coefficients
}

plot(coeffs, xlab="betahat1", ylab="betahat2") #create the scatterplot
```



1 (b) (ii)

```r
betahat1.mean <- mean(coeffs[,1])
betahat2.mean <- mean(coeffs[,2])

betahat1.mean
```

```
## [1] 12.18651
```

```r
betahat2.mean
```

```
## [1] 0.028471
```

The sample mean for $\hat{\beta}_1^{(j)}$ is 12.18651 and the sample mean for $\hat{\beta}_2^{(j)}$ is 0.028471.

1 (b) (iii)

```r
betahat1.var <- var(coeffs[,1])
betahat2.var <- var(coeffs[,2])

betahat1.var
```

```
## [1] 3.773097
```

```r
betahat2.var
```

```
## [1] 0.001861814
```

The sample variance for $\hat{\beta}_1^{(j)}$ is 3.773097 and the sample variance for $\hat{\beta}_2^{(j)}$ is 0.001861814.

1 (b) (iv)

```r
betahat.cov <-cov(coeffs[,1],coeffs[,2])

betahat.cov
```

```
## [1] -0.03864999
```

The sample correlation between $\hat{\beta}_1^{(j)}$ and $\hat{\beta}_2^{(j)}$ is -0.03864999.

1 (c) (i)

```r
#prepare data for the model
d1 <- list (visits = log(plaindata),
            year = xs,
            mubeta0 = c(0,0),
            Sigmamubetainv = rbind(c(0.000001, 0),
                                   c(0, 0.000001)),
            Sigma0 = rbind(c(100, 0),
                           c(0, 0.1)))
#initialize chains
inits1 <- list(list(sigmasqyinv=10, mubeta=c(1000,1000),
                    Sigmabetainv = rbind(c(100,0),
                                         c(0,100)),
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123456),
               list(sigmasqyinv=0.001, mubeta=c(-1000,1000),
                    Sigmabetainv = rbind(c(100,0),
                                         c(0,100)),
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123455),
               list(sigmasqyinv=10, mubeta=c(1000,-1000),
                    Sigmabetainv = rbind(c(0.001,0),
                                         c(0,0.001)),
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123454),
               list(sigmasqyinv=0.001, mubeta=c(-1000,-1000),
                    Sigmabetainv = rbind(c(0.001,0),
                                         c(0,0.001)),
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123453))


library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```r
#print the model out
cat (readLines('./1ci.bug'), sep= '\n')
```

```
## data {
##   dimY <- dim(visits)
##   yearcent <- year - mean(year)
## }
## model {
##   for (j in 1:dimY[1]) {
##     for (i in 1:dimY[2]) {
##       visits[j,i] ~ dnorm (beta[1,j]+beta[2,j]*yearcent[i], sigmasqyinv)
##     }
##     beta[1:2,j] ~ dmnorm (mubeta, Sigmabetainv)
##   }
##
##   mubeta ~ dmnorm(mubeta0, Sigmamubetainv)
##   Sigmabetainv ~ dwish(2*Sigma0, 2)
##   sigmasqyinv ~ dgamma(0.0001, 0.0001)
##
##   Sigmabeta <- inverse(Sigmabetainv)
##   rho <- Sigmabeta[1,2]/sqrt(Sigmabeta[1,1]*Sigmabeta[2,2])
##   sigmasqy <- 1/sigmasqyinv
## }
```

```r
#create model
m1 <- jags.model("./1ci.bug", d1, inits1, n.chains=4, n.adapt=1000)
```

```
## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 420
##    Unobserved stochastic nodes: 33
##    Total graph size: 1408
##
## Initializing model
```

```r
update(m1, 23000) # burn-in

x1 <- coda.samples(m1, c("mubeta", "Sigmabeta", "sigmasqy", "rho"), n.iter=4000) #get samples

gelman.diag(x1, autoburnin=FALSE, multivariate=FALSE) #shows convergence if < 1.1
```

```
## Potential scale reduction factors:
##
##                Point est. Upper C.I.
## Sigmabeta[1,1]          1          1
## Sigmabeta[2,1]          1          1
## Sigmabeta[1,2]          1          1
## Sigmabeta[2,2]          1          1
## mubeta[1]               1          1
## mubeta[2]               1          1
## rho                     1          1
## sigmasqy                1          1
```

```r
effectiveSize(x1) #check effective size > 4000
```

```
## Sigmabeta[1,1] Sigmabeta[2,1] Sigmabeta[1,2] Sigmabeta[2,2]      mubeta[1]
##       15227.22       14566.34       14566.34       15052.33       16287.62
##      mubeta[2]            rho       sigmasqy
##       15428.95       14486.81       12033.41
```

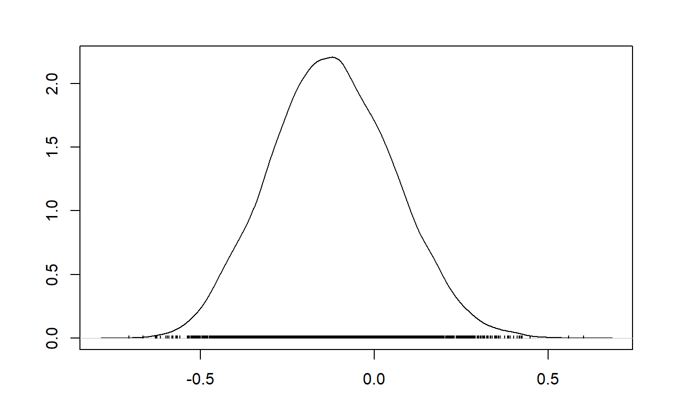
1 (c) (ii)

```r
summary(x1)
```

```
##
## Iterations = 23001:27000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                     Mean       SD  Naive SE Time-series SE
## Sigmabeta[1,1] 11.050382 3.072853 2.429e-02      2.494e-02
## Sigmabeta[2,1] -0.039200 0.062032 4.904e-04      5.150e-04
## Sigmabeta[1,2] -0.039200 0.062032 4.904e-04      5.150e-04
## Sigmabeta[2,2]  0.009202 0.002557 2.021e-05      2.086e-05
## mubeta[1]      12.191982 0.610129 4.823e-03      4.781e-03
## mubeta[2]       0.028392 0.017725 1.401e-04      1.429e-04
## rho            -0.120954 0.177263 1.401e-03      1.475e-03
## sigmasqy        0.056392 0.004254 3.363e-05      3.880e-05
##
## 2. Quantiles for each variable:
##
##                     2.5%       25%       50%        75%    97.5%
## Sigmabeta[1,1]  6.527671  8.873499 10.561764  1.262e+01 18.54533
## Sigmabeta[2,1] -0.173227 -0.074900 -0.036508 -3.883e-06  0.07667
## Sigmabeta[1,2] -0.173227 -0.074900 -0.036508 -3.883e-06  0.07667
## Sigmabeta[2,2]  0.005458  0.007368  0.008776  1.059e-02  0.01527
## mubeta[1]      10.991060 11.783992 12.185121  1.260e+01 13.38896
## mubeta[2]      -0.006027  0.016660  0.028311  3.997e-02  0.06392
## rho            -0.455901 -0.242696 -0.123776 -1.981e-05  0.23164
## sigmasqy        0.048708  0.053439  0.056161  5.912e-02  0.06538
```

1 (c) (iii)

```r
densplot(x1[,c("rho")])
```



A 95% central posterior interval for ρ is $(-0.455901, 0.23164)$

1 (c) (iv)

```r
mean(as.matrix(x1[,c("rho")])<0) #posterior probability for rho < 0
```

```
## [1] 0.75
```

```r
mean(as.matrix(x1[,c("rho")])<0)/mean(as.matrix(x1[,c("rho")])>0) #calculate Bayes factor as posterior/prior
```

```
## [1] 3
```

The posterior probability that $\rho < 0$ is 0.75, and the Bayes factor favoring $\rho < 0$ over $\rho > 0$ is 3 which means the data evidence is between barely mentionable and positive.

1 (c) (v)

```r
# reading the quantiles for mubeta[2] from 1cii
exp(13*-0.006027)
```

```
## [1] 0.9246398
```

```r
exp(13*0.06392)
```

```
## [1] 2.295521
```

The 95% central posterior interval for $e^{13\mu_{\beta_2}}$ is $(0.9246398, 2.295521)$

1 (c) (vi)

```r
dic.samples(m1, 100000)
```

```
## Mean deviance:  -17.54
## penalty 60.51
## Penalized deviance: 42.97
```

The effective number of parameters is \$60.51 \$ and the Plummer's DIC is 42.97.

1 (d) (i)

I assumed that we can take X to just be a vector consisting of the i different years and the so the same X is used for each y.

1 (d) (ii)

```r
#prepare data for the model
d2 <- list (visits = log(plaindata),
            year = xs)

#initialize chains, adding initial values for each sigmabeta and mubeta
inits2 <- list(list(sigmasqyinv=10, mubeta1=1000, mubeta2=1000,
                    sigmabeta1=1000, sigmabeta2=1000,
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123456),
               list(sigmasqyinv=0.001, mubeta1=-1000, mubeta2=1000,
                    sigmabeta1=1000, sigmabeta2=1000,
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123457),
               list(sigmasqyinv=10, mubeta1=1000, mubeta2=-1000,
                    sigmabeta1=0.001, sigmabeta2=0.001,
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123458),
               list(sigmasqyinv=0.001, mubeta1=--1000, mubeta2=-1000,
                    sigmabeta1=0.001, sigmabeta2=0.001,
                    .RNG.name="base::Wichmann-Hill", .RNG.seed=123459))


library(rjags)

#print the model out
cat (readLines('./1dii.bug'), sep= '\n')
```

```
## data {
##   dimY <- dim(visits)
##   yearcent <- year - mean(year)
## }
## model {
##   for (j in 1:dimY[1]) {
##     for (i in 1:dimY[2]) {
##       visits[j,i] ~ dnorm (beta[1,j]+beta[2,j]*yearcent[i], sigmasqyinv)
##     }
##     beta[1,j] ~ dnorm (mubeta1, sigmabeta1sqinv)
##     beta[2,j] ~ dnorm (mubeta2, sigmabeta2sqinv)
##   }
##
##   mubeta1 ~ dnorm(0, 0.000001)
##   mubeta2 ~ dnorm(0, 0.000001)
##   sigmabeta1 ~ dunif(0, 1000)
##   sigmabeta2 ~ dunif(0, 1000)
##   sigmasqyinv ~ dgamma(0.0001, 0.0001)
##
##   sigmabeta1sq <- sigmabeta1^2
##   sigmabeta2sq <- sigmabeta2^2
##   sigmabeta1sqinv <- 1/sigmabeta1sq
##   sigmabeta2sqinv <- 1/sigmabeta2sq
##   sigmasqy <- 1/sigmasqyinv
## }
```

```r
#create model
m2 <- jags.model("./1dii.bug", d2, inits2, n.chains=4, n.adapt=2000)
```

```
## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 420
##    Unobserved stochastic nodes: 65
##    Total graph size: 1366
##
## Initializing model
```

```r
update(m2, 80000) # burn-in

x2 <- coda.samples(m2, c("mubeta1", "mubeta2", "sigmabeta1sq", "sigmabeta2sq", "sigmasqy"), n.iter=4000) #get samples

gelman.diag(x2, autoburnin=FALSE, multivariate=FALSE) #show convergence
```

```
## Potential scale reduction factors:
##
##              Point est. Upper C.I.
## mubeta1            1.00       1.00
## mubeta2            1.00       1.00
## sigmabeta1sq       1.09       1.27
## sigmabeta2sq       1.00       1.00
## sigmasqy           1.00       1.00
```

```r
effectiveSize(x2) #check effective size > 4000
```

```
##      mubeta1      mubeta2 sigmabeta1sq sigmabeta2sq     sigmasqy
##    15375.126    12694.756     4217.260     6447.885    11542.710
```

1 (d) (iii)

```r
summary(x2)
```

```
##
## Iterations = 82001:86000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                   Mean        SD  Naive SE Time-series SE
## mubeta1      12.190152 0.3632376 2.872e-03      2.929e-03
## mubeta2       0.028480 0.0082165 6.496e-05      7.296e-05
## sigmabeta1sq  4.012323 1.1419435 9.028e-03      7.755e-02
## sigmabeta2sq  0.001809 0.0005928 4.687e-06      7.959e-06
## sigmasqy      0.056390 0.0042306 3.345e-05      3.949e-05
##
## 2. Quantiles for each variable:
##
##                   2.5%      25%       50%       75%     97.5%
## mubeta1      1.147e+01 11.95215 12.192794 12.428588 12.903766
## mubeta2      1.239e-02  0.02301  0.028457  0.033900  0.044543
## sigmabeta1sq 2.531e+00  3.29150  3.742768  4.485809  6.897780
## sigmabeta2sq 9.538e-04  0.00139  0.001705  0.002114  0.003272
## sigmasqy     4.870e-02  0.05345  0.056167  0.059113  0.065222
```

1 (d) (iv)

```r
# reading the quantiles for mubeta2 from 1diii
exp(13*0.01239)
```

```
## [1] 1.174767
```

```r
exp(13*0.044543)
```

```
## [1] 1.784359
```

An approximate 95% central posterior interval is $(1.174767, 1.784359)$ which is a much more restrictive interval than the first model.

1 (d) (v)

```r
dic.samples(m2, 100000)
```

```
## Mean deviance:  -16.63
## penalty 57.82
## Penalized deviance: 41.19
```

The effective number of parameters is 57.82 and the Plummer's DIC is 41.19.

1 (d) (vi)

The Plummer's DIC value for the first model was 42.97, which is higher than this model's DIC value. Therefore, this model would be preferred, althought not by much.