BDA - Project Work

Jacopo Losi, Nicola Saljoughi

Contents


```
mydata <- data
mydata$Season <- data$Month
mydata$Month = NULL


# Hospitalised

indexHosp   <- which(data$Hospitalised == 'yes')
indexNoHosp <- which(data$Hospitalised == 'no')

mydata$Hospitalised[indexHosp]   <- 1    # 1 --> yes
mydata$Hospitalised[indexNoHosp] <- 0    # 0 --> no


# Died

indexDied   <- which(data$Died == 'yes')
indexNoDied <- which(data$Died == 'no')

mydata$Died[indexDied]   <- 1    # 1 --> yes
mydata$Died[indexNoDied] <- 0    # 0 --> no


# Urban

indexUrban   <- which(data$Urban == 'yes')
indexNoUrban <- which(data$Urban == 'no')

mydata$Urban[indexUrban]   <- 1    # 1 --> yes
mydata$Urban[indexNoUrban] <- 0    # 0 --> no
```

```r
# Season

indexSpring <- which(data$Month >= 3 & data$Month <= 5)
indexSummer <- which(data$Month >= 6 & data$Month <= 8)
indexAutumn <- which(data$Month >= 9 & data$Month <= 11)
indexWinter <- which(data$Month == 12 | data$Month <= 2)

mydata$Season[indexSpring] <- 1  # 1 --> Spring
mydata$Season[indexSummer] <- 2  # 2 --> Summer
mydata$Season[indexAutumn] <- 3  # 3 --> Autumn
mydata$Season[indexWinter] <- 4  # 4 --> Winter


# Sex

indexMale   <- which(data$Sex == 'male')
indexFemale <- which(data$Sex == 'female')

mydata$Sex[indexMale]   <- 1    # 1 --> male
mydata$Sex[indexFemale] <- 0    # 0 --> female


# Age

indexAgeOne   <- which(data$Age <= 34)
indexAgeTwo   <- which(data$Age >= 35 & data$Age <= 49)
indexAgeThree <- which(data$Age >= 50 & data$Age <= 64)
indexAgeFour  <- which(data$Age >= 65)

mydata$Age[indexAgeOne]   <- 1   # 1 --> <34
mydata$Age[indexAgeTwo]   <- 2   # 2 --> 35-49
mydata$Age[indexAgeThree] <- 3   # 3 --> 50-64
mydata$Age[indexAgeFour]  <- 4   # 4 --> >65


# Education

indexEduZero  <- which(data$Education == 'iliterate')
indexEduOne   <- which(data$Education == 'primary')
indexEduTwo   <- which(data$Education == 'Secondary')
indexEduThree <- which(data$Education == 'Tertiary')

mydata$Education[indexEduZero]   <- 0   # 0 --> iliterate
mydata$Education[indexEduOne]    <- 1   # 1 --> primary
mydata$Education[indexEduTwo]    <- 2   # 2 --> Secondary
mydata$Education[indexEduThree]  <- 3   # 3 --> Tertiary


# Occupation

indexFarm   <- which(data$Occupation == 'farming')
indexNoFarm <- which(data$Occupation != 'farming')
```

```r
mydata$Occupation[indexFarm]   <- 1    # 1 --> farming
mydata$Occupation[indexNoFarm] <- 0    # 0 --> non farming


# Method

indexPesticide <- which(data$method == 'Pesticide')
indexPoison    <- which(data$method == 'Other poison')
indexHanging    <- which(data$method == 'hanging')
indexOthers    <- which(data$method != 'Pesticide' &
                        data$method != 'Other poison' &
                        data$method != 'hanging')

mydata$method[indexPesticide] <- 1 # 1 --> Pesticide
mydata$method[indexPoison]    <- 2 # 2 --> Other poison
mydata$method[indexHanging]   <- 3 # 3 --> hanging
mydata$method[indexOthers]    <- 4 # 4 --> All others
```

Introduction

Objective

The objective of the study is to estimate the incidence of seriours suicide attempts (SSAs), defined as suicide attempts resulting in either death or hospitalization, and to analyse the factors associated with fatality among the attempters.

Data

The data set is constituted by 2571 observations of 11 variables:

- `Person_ID`: ID number, $1, ..., 2571$
- `Hospitalised`: *yes* or *no*
- `Died`: *yes* or *no*
- `Urban`: *yes*, *no* or *unknown*
- `Year`: 2009, 2010 or 2011
- `Month`: $1, ..., 12$
- `Sex`: *female* or *male*
- `Age`: years
- `Education`: *iliterate*, *primary*, *Secondary*, *Tertiary* or *unknown*
- `Occupation`: one of ten categories
- `method`: one of nine methods

Source

Sun J, Guo X, Zhang J, Wang M, Jia C, Xu A (2015) "Incidence and fatality of serious suicide attempts in a predominantly rural population in Shandong, China: a public health surveillance study," BMJ Open 5(2): e006762. https://doi.org/10.1136/bmjopen-2014-006762

Data downloaded via Dryad Digital Repository. https://doi.org/10.5061/dryad.r0v35

Analysis

```
rural_men   <- subset(data, data$Sex=="male" & data$Urban=="no")
rural_women <- subset(data, data$Sex=="female" & data$Urban=="no")
urban_men   <- subset(data, data$Sex=="male" & data$Urban=="yes")
urban_women <- subset(data, data$Sex=="female" & data$Urban=="yes")
```

```
str(mydata)
```

```
## 'data.frame':    2571 obs. of  11 variables:
##  $ Person_ID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Hospitalised: chr  "1" "0" "0" "0" ...
##  $ Died        : chr  "0" "1" "1" "1" ...
##  $ Urban       : chr  "0" "0" "0" "0" ...
##  $ Year        : int  2010 2009 2010 2011 2009 2009 2010 2010 2010 2011 ...
##  $ Sex         : chr  "0" "1" "1" "1" ...
##  $ Age         : num  2 4 3 4 3 3 4 3 4 1 ...
##  $ Education   : chr  "2" "1" "1" "1" ...
##  $ Occupation  : chr  "0" "1" "1" "1" ...
##  $ method      : chr  "2" "3" "3" "3" ...
##  $ Season      : num  4 1 4 4 2 3 4 3 2 4 ...
```

	All SAAs	Hospitalised and survived	Hospitalised but died	Total SSA hospitalisations	SSA deaths without hospitalisation	Total SSA deaths
Urban						
Female	149	99	18	117	32	50
Male	128	65	17	82	46	63
Both	277	164	35	199	78	113
Rural						
Female	1134	598	100	698	436	536
Male	1079	474	103	577	502	605
Both	2213	1072	203	1275	938	1141
Total						
Female	1328	741	118	859	469	587
Male	1243	574	120	694	549	669
Both	2571	1315	238	1553	1018	1256

```
## Remove unknown labels

indexUnknw1 <- which(mydata$Education == 'unknown')
mydata <- mydata[-indexUnknw1,]
indexUnkn <- which(mydata$Urban == 'unknown')
mydata <- mydata[-indexUnkn,]
```

Then in this phase, in which we are working on testing different models, it is worth to take only some random samples from the data. As a matter of fact, the dataset that we have is big and thus the computation on the whole dataset will take a lot of time.

Therefore, we will proceed as follows: * we will generate a vector of 50 random number taken from our dataset; * we will test the models with this data, that are sufficient for not loosing in generality; * we will run the final model on the whole dataset.

```
random_index <- sample(mydata$Person_ID, size = 50, replace = TRUE)

data_reduced <- mydata[random_index, ]
data_reduced <- na.omit(data_reduced)
```

```
## Create Stan data
dat <- list(N        = nrow(data_reduced),
            p        = ncol(data_reduced) - 2,
            died     = as.numeric(data_reduced$Died),
            urban    = as.numeric(data_reduced$Urban),
            year     = as.numeric(data_reduced$Year),
            season   = as.numeric(data_reduced$Season),
            sex      = as.numeric(data_reduced$Sex),
            age      = as.numeric(data_reduced$Age),
            edu      = as.numeric(data_reduced$Education),
            job      = as.numeric(data_reduced$Occupation),
            method   = as.numeric(data_reduced$method))

## Load Stan file
fileName <- "./logistic_regression_model.stan"
stan_code <- readChar(fileName, file.info(fileName)$size)
cat(stan_code)
```

```
## data {
##   // Define variables in data
##   // Number of observations (an integer)
##   int<lower=0> N;
##
##   // Number of parameters
##   int<lower=0> p;
##
##   // Variables
##   int  died[N];
##   int<lower=0>  year[N];
##   int<lower=0>  urban[N];
##   int<lower=0>  season[N];
##   int<lower=0>  sex[N];
##   int<lower=0>  age[N];
##   int<lower=0>  edu[N];
##   int<lower=0>  job[N];
##   int<lower=0>  method[N];
## }
##
## parameters {
##   // Define parameters to estimate
##   real beta[p];
```

```
## }
##
## transformed parameters  {
##   // Probability trasformation from linear predictor
##   real<lower=0> odds[N];
##   real<lower=0, upper=1> prob[N];
##   for (i in 1:N) {
##     odds[i] = exp(beta[1] + beta[2]*year[i]  + beta[3]*urban[i] +
##                             beta[4]*season[i] + beta[5]*sex[i]  +
##                             beta[6]*age[i]    + beta[7]*edu[i]  +
##                             beta[8]*job[i]    + beta[9]*method[i] );
##     prob[i] = odds[i] / (odds[i] + 1);
##   }
## }
##
## model {
##   // Prior part of Bayesian inference (flat if unspecified)
##
##   // Likelihood part of Bayesian inference
##     died ~ bernoulli(prob);
## }
```

```r
# Run Stan
resStan <- stan(model_code = stan_code,
                data = dat,
                chains = 3,
                iter = 1000,
                warmup = 500,
                thin = 10)
```

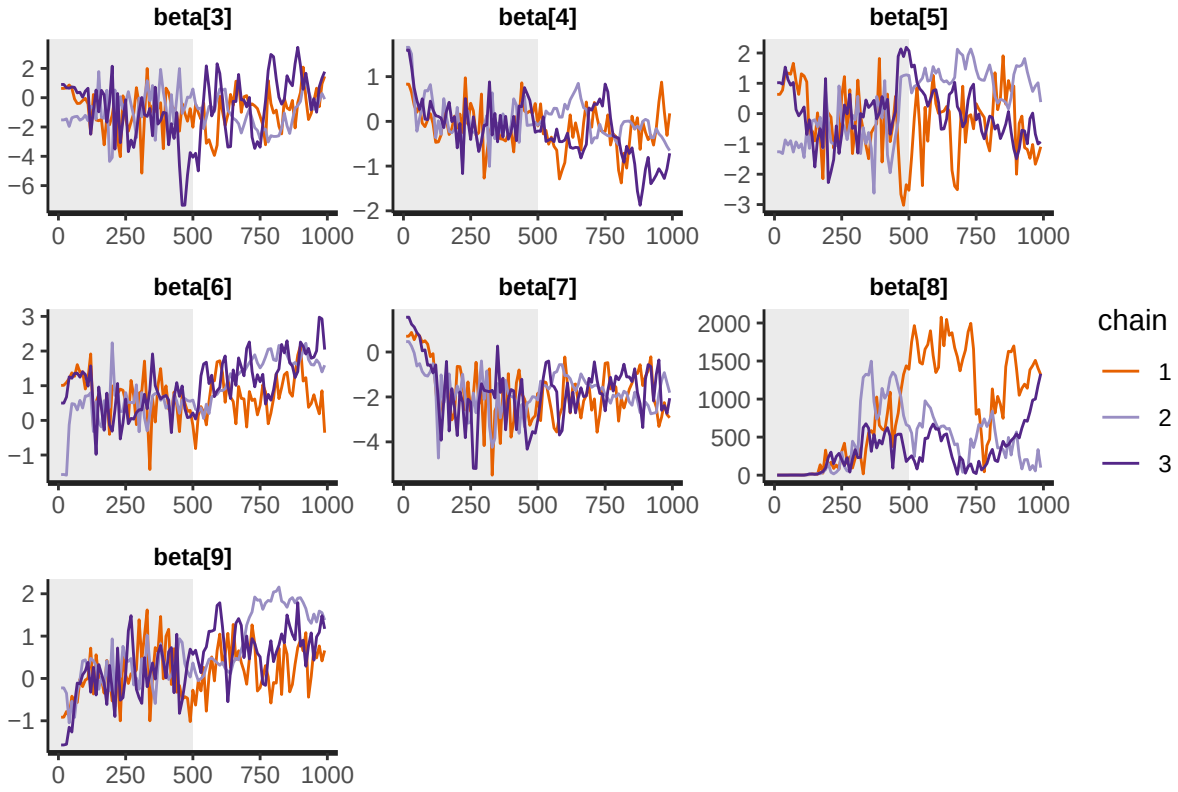
```
##
## SAMPLING FOR MODEL '44efd1e4898e49d7c3da763fa46eaad0' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 18.29 seconds (Warm-up)
## Chain 1:                15.054 seconds (Sampling)
## Chain 1:                33.344 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '44efd1e4898e49d7c3da763fa46eaad0' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
```

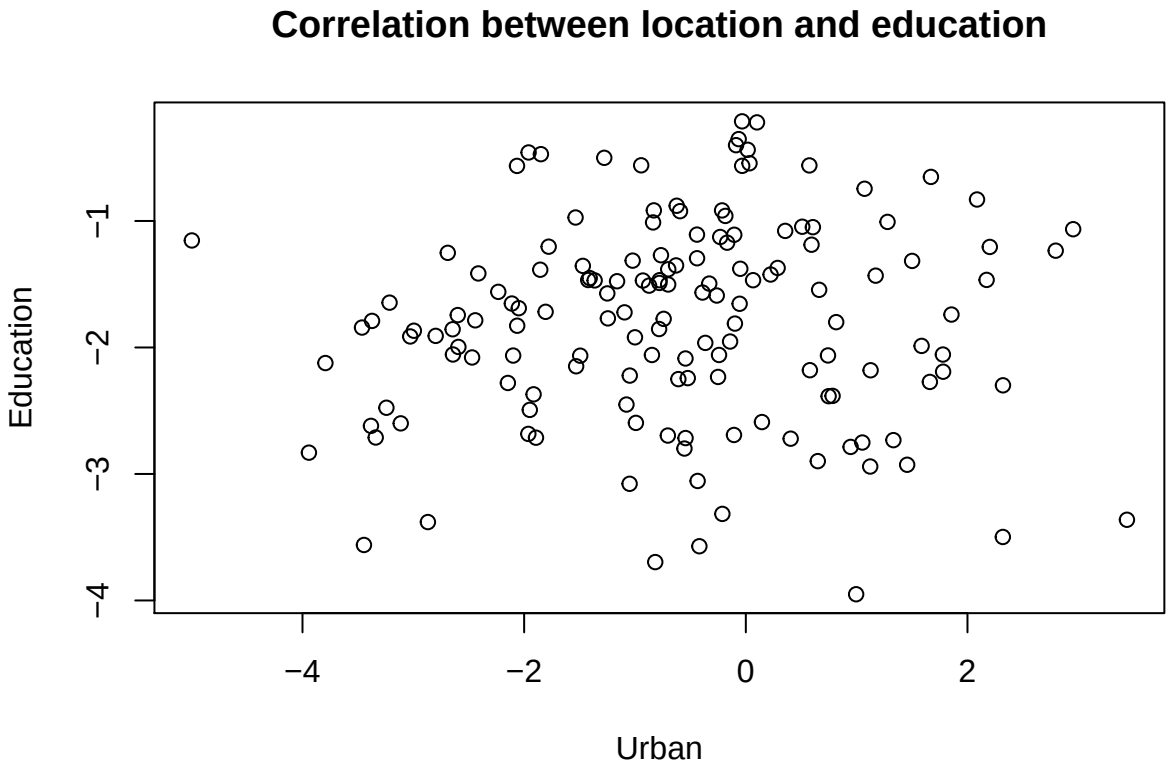
```
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 11.872 seconds (Warm-up)
## Chain 2:                16.367 seconds (Sampling)
## Chain 2:                28.239 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '44efd1e4898e49d7c3da763fa46eaad0' NOW (CHAIN 3).
## Chain 3: Rejecting initial value:
## Chain 3:   Error evaluating the log probability at the initial value.
## Chain 3: Exception: validate transformed params: prob[i_0__] is nan, but must be greater than or equa
##
## Chain 3: Rejecting initial value:
## Chain 3:   Log probability evaluates to log(0), i.e. negative infinity.
```

```
## Chain 3:   Stan can't start sampling from this initial value.
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 13.016 seconds (Warm-up)
## Chain 3:                15.99 seconds (Sampling)
## Chain 3:                29.006 seconds (Total)
## Chain 3:

## Warning: There were 148 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is 2.42, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```r
# Show traceplot
traceplot(resStan, pars = c('beta[3]','beta[4]', 'beta[5]',
                            'beta[6]', 'beta[7]', 'beta[8]',
                            'beta[9]'), inc_warmup = TRUE)
```




```
# Generate some scatter plots in order to see the correlations between parameters
plot(extract(resStan)$beta[,3], extract(resStan)$beta[,7], main="Correlation between location and educa
     xlab="Urban", ylab="Education")
```




```
# qplot(extract(resStan)$beta[,3], bins = 30, geom = 'histogram', main='Histogram for Urban',
#                         xlab = 'weigths for the coefficient', ylab = 'Occurrences',
#                         col = I('red'))

# overlay histogram, density and show the mean value
# The plots of the most interesting parameters are presented.
# Using the mean value it could be interesting to understand
# which weaknly informative priors can be designed

plot_1 <- qplot(extract(resStan)$beta[,3], geom = 'blank', xlab = 'Values of weigth', ylab = 'Occurences
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStan)$beta[,3])), col=I('yellow'), linetype="dashed", size=

plot_2 <- qplot(extract(resStan)$beta[,5], geom = 'blank', xlab = 'Values of weigth', ylab = 'Occurences
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStan)$beta[,5])), col=I('yellow'), linetype="dashed", size=

plot_3 <- qplot(extract(resStan)$beta[,6], geom = 'blank', xlab = 'Values of weigth', ylab = 'Occurences
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStan)$beta[,6])), col=I('yellow'), linetype="dashed", size=

plot_4 <- qplot(extract(resStan)$beta[,7], geom = 'blank', xlab = 'Values of weigth', ylab = 'Occurences
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
```

```
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStan)$beta[,7])), col=I('yellow'), linetype="dashed", size=

plot_5 <- qplot(extract(resStan)$beta[,8], geom = 'blank', xlab = 'Values of weigth', ylab = 'Occurences
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStan)$beta[,8])), col=I('yellow'), linetype="dashed", size=

ggplot2.multiplot(plot_1,plot_2,plot_3,plot_4, plot_5, cols=3)
```

From the analysis done above, and especially looking at the histogram, it is clear that the most important parameters that count in our analysis are: the fact that the people come from urbn or rural areas, then their education and occupation. As a matter of fact, the mean and the maximum values of the coeffcient related to those paramters have the bigger magnitude. This means that those parameters are weighted more in the multi regression function in the model.

Therefore, for further analysis, it will be good to develop specific analysis using only these parameters, in order to have a more precise evalution considering only the most relevant parameters.

Frequentist approach

```
outcomeModel <- glm(as.numeric(Died) ~ as.numeric(Urban) +
                                       as.numeric(Year) +
                                       as.numeric(Season) +
```

```
                                      as.numeric(Sex) +
                                      as.numeric(Age) +
                                      as.numeric(Education) +
                                      as.numeric(Occupation) +
                                      as.numeric(method), data = mydata,
                    family = binomial(link = "logit"))
summary(outcomeModel)
```

```
##
## Call:
## glm(formula = as.numeric(Died) ~ as.numeric(Urban) + as.numeric(Year) +
##     as.numeric(Season) + as.numeric(Sex) + as.numeric(Age) +
##     as.numeric(Education) + as.numeric(Occupation) + as.numeric(method),
##     family = binomial(link = "logit"), data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3828  -0.8351   0.3501   0.8233   2.5409
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -5.858e+02  1.245e+02  -4.704 2.56e-06 ***
## as.numeric(Urban)       2.961e-01  1.611e-01   1.837   0.0662 .
## as.numeric(Year)        2.916e-01  6.197e-02   4.706 2.53e-06 ***
## as.numeric(Season)      8.641e-03  4.488e-02   0.193   0.8473
## as.numeric(Sex)         4.243e-01  9.900e-02   4.286 1.82e-05 ***
## as.numeric(Age)         3.317e-01  5.295e-02   6.265 3.73e-10 ***
## as.numeric(Education)  -1.248e+00  8.083e-02 -15.443  < 2e-16 ***
## as.numeric(Occupation)  5.188e-01  1.316e-01   3.942 8.07e-05 ***
## as.numeric(method)     -5.107e-02  4.509e-02  -1.133   0.2574
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3344.4  on 2413  degrees of freedom
## Residual deviance: 2560.3  on 2405  degrees of freedom
## AIC: 2578.3
##
## Number of Fisher Scoring iterations: 4
```

Comparison

```
## Bayesian
print(resStan, pars = c("beta"))
```

```
## Inference for Stan model: 44efd1e4898e49d7c3da763fa46eaad0.
## 3 chains, each with iter=1000; warmup=500; thin=10;
## post-warmup draws per chain=50, total post-warmup draws=150.
##
##             mean se_mean      sd     2.5%      25%      50%      75%
```

```
## beta[1] -4382.24  552.02 1235.83 -6904.20 -5003.64 -4349.78 -3638.09
## beta[2]     1.81    0.45    0.76     0.14     1.44     1.88     2.27
## beta[3]    -0.63    0.31    1.57    -3.45    -1.80    -0.60     0.34
## beta[4]    -0.28    0.13    0.52    -1.38    -0.56    -0.29     0.08
## beta[5]     0.26    0.53    1.01    -1.90    -0.45     0.34     1.05
## beta[6]     1.19    0.25    0.69    -0.15     0.68     1.31     1.68
## beta[7]    -1.79    0.11    0.79    -3.52    -2.27    -1.74    -1.26
## beta[8]   750.86  367.40  566.10    38.95   280.63   606.12  1141.90
## beta[9]     0.78    0.26    0.68    -0.47     0.31     0.70     1.30
##            97.5% n_eff Rhat
## beta[1] -1914.31     5 1.81
## beta[2]     3.13     3 2.64
## beta[3]     2.32    26 1.14
## beta[4]     0.72    15 1.24
## beta[5]     1.93     4 1.49
## beta[6]     2.26     7 1.32
## beta[7]    -0.43    50 1.07
## beta[8]  1916.12     2 1.99
## beta[9]     1.91     7 1.45
##
## Samples were drawn using NUTS(diag_e) at Thu Dec 05 19:09:17 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
## Frequentist
tableone::ShowRegTable(outcomeModel, exp = FALSE)
```

```
##                        coef [confint]             p
## (Intercept)            -585.81 [-830.91, -342.52] <0.001
## as.numeric(Urban)         0.30 [-0.02, 0.61]       0.066
## as.numeric(Year)          0.29 [0.17, 0.41]       <0.001
## as.numeric(Season)        0.01 [-0.08, 0.10]       0.847
## as.numeric(Sex)           0.42 [0.23, 0.62]       <0.001
## as.numeric(Age)           0.33 [0.23, 0.44]       <0.001
## as.numeric(Education)    -1.25 [-1.41, -1.09]     <0.001
## as.numeric(Occupation)    0.52 [0.26, 0.78]       <0.001
## as.numeric(method)       -0.05 [-0.14, 0.04]       0.257
```

Same clustering on the data

Let us try to cluster the data using the specific year in order to do a prediction on the following year

```
indexYear2009 <- which(mydata$Year == 2009)
data_year_2009 <- mydata[indexYear2009,]

indexYear2010 <- which(mydata$Year == 2010)
data_year_2010 <- mydata[indexYear2010,]

indexYear2011 <- which(mydata$Year == 2011)
data_year_2011 <- mydata[indexYear2011,]
```

Conclusions