BDA - Project Work

Jacopo Losi, Nicola Saljoughi

Contents

```r
datFile <- "suicide attempt data_2.csv"
datCsv <- read.csv(datFile, stringsAsFactors=FALSE)
datSet <- as.data.frame(datCsv)

datSet$Season <- datSet$Month
datSet$Month = NULL

## Remove unknown labels

indexUnkn_1 <- which(datSet$Education == 'unknown')
indexUnkn_2 <- which(datSet$Urban == 'unknown')
indexUnkn_3 <- which(datSet$Occupation == 'others/unknown')
datSet <- datSet[-c(indexUnkn_1, indexUnkn_2,indexUnkn_3),]
```

```
# Hospitalised
indexHosp   <- which(datSet$Hospitalised == 'yes')
indexNoHosp <- which(datSet$Hospitalised == 'no')
datSet$Hospitalised[indexHosp]   <- 1    # 1 --> yes
datSet$Hospitalised[indexNoHosp] <- 0    # 0 --> no


# Died
indexDied   <- which(datSet$Died == 'yes')
indexNoDied <- which(datSet$Died == 'no')
datSet$Died[indexDied]   <- 1    # 1 --> yes
datSet$Died[indexNoDied] <- 0    # 0 --> no


# Urban
indexUrban   <- which(datSet$Urban == 'yes')
indexNoUrban <- which(datSet$Urban == 'no')
datSet$Urban[indexUrban]   <- 1    # 1 --> yes
datSet$Urban[indexNoUrban] <- 0    # 0 --> no

#Year
indexYear2009 <- which(datSet$Year == 2009)
indexYear2010 <- which(datSet$Year == 2010)
indexYear2011 <- which(datSet$Year == 2011)
datSet$Year[indexYear2009] <- 1    # 1 --> 2009
datSet$Year[indexYear2010] <- 2    # 2 --> 2010
datSet$Year[indexYear2011] <- 3    # 3 --> 2011

# Sex
indexMale   <- which(datSet$Sex == 'male')
indexFemale <- which(datSet$Sex == 'female')
datSet$Sex[indexMale]   <- 1    # 1 --> male
datSet$Sex[indexFemale] <- 0    # 0 --> female

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

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


# Occupation
indexUnEmpl <- which(datSet$Occupation == 'unemployed')
indexFarm   <- which(datSet$Occupation == 'farming')
indexProf   <- which(datSet$Occupation == 'business/service' | datSet$Occupation == 'professional' | da

datSet$Occupation[indexUnEmpl]                             <- 0   # 0 --> unemployed
datSet$Occupation[indexFarm]                               <- 1   # 1 --> farming
```

```r
datSet$Occupation[indexProf]                        <- 2   # 2 --> professional and worker
datSet$Occupation[-c(indexUnEmpl, indexFarm, indexProf)]  <- 3   # 3 --> others

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

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

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

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

datSetCluster <- datSet

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

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

Introduction

This project is based on a study carried out in 2015 by a group of researchers to estimate the incidence of serious suicide attempts in Shandong, China, and to examine the factors associated with fatality among the attempters.
We have chosen to examine a dataset on suicides because it is a really important but often underconsidered problem in today's society. Not only this problem reflects a larger problem in a country societal system but it can also be a burden for hospital resources. We think that by being able to talk about it more openly and by truly trying to estimate its size and impact we can start to understand where the causes are rooted and what can be done to fight it.
We invite the reader to check the source section to further read about the setting and results of the named paper.
In this report we will carry out our analysis following the bayesian approach. Since also the frequentist approach was covered during lecture, we though that it was meaningful to compare the two at the begininning of the analysis.
Adopting then the bayesian approach we will first develop a multiple logistic regression model using all the variables, after that we will do variable selection to determine which are the most influential factors and develop a second multiple logistic regression model using the selected variables. After that, we will assess the convergence and efficiency of the models, do posterior predictive checking and compare the models. To conclude we carry out a prediction on the age of the attempters and eventually answer our analysis problem.

Analysis Problem

The objective of the project is to use the bayesian approach to develop models to evaluate the most influential factors related to serious suicide attempts (SSAs, defined as suicide attempts resulting in either death or hospitalisation) and being able to make predictions on the age of the attempters.

Data

Data from two independent health surveillance systems were linked, constituted by records of suicide deaths and hospitalisations that occured among residents in selected countries during 2009-2011.
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

It is important to notice that the population in the study is predominantly rural and that the limitation of the study is that the incidence estimates are likely to be underestimated due to underreporting in both surveillance systems.

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

Model description

In order to evaluate the factors which influence the probability of SSA the most it is an obvious chioice to develop a multiple logistic regression model.

As already said, at first we evaluated a comparison between the frequentist and the bayesian approach, in order to provide an analysis of the dataset with both models, being the two approaches discussed during the lectures.

Thus, the two model follow below.

Frequentist approach

```
freqModel <- 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 = datSetCluster,
                family = binomial(link = "logit"))
summary(freqModel)
```

```
##
## 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 = datSetCluster)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4022  -0.8521   0.3783   0.8193   2.4227
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)              0.46704    0.34494   1.354 0.175746
## as.numeric(Urban)        0.24135    0.16531   1.460 0.144295
## as.numeric(Year)         0.30554    0.06300   4.850 1.24e-06 ***
## as.numeric(Season)       0.01583    0.04573   0.346 0.729242
## as.numeric(Sex)          0.39150    0.10135   3.863 0.000112 ***
## as.numeric(Age)          0.34280    0.05350   6.408 1.47e-10 ***
## as.numeric(Education)  -1.25613    0.08219 -15.283  < 2e-16 ***
## as.numeric(Occupation) -0.21951    0.07595  -2.890 0.003852 **
## as.numeric(method)     -0.04884    0.04582  -1.066 0.286438
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3221.6  on 2329  degrees of freedom
## Residual deviance: 2474.8  on 2321  degrees of freedom
## AIC: 2492.8
##
## Number of Fisher Scoring iterations: 4
```

Bayesian approach using Stan

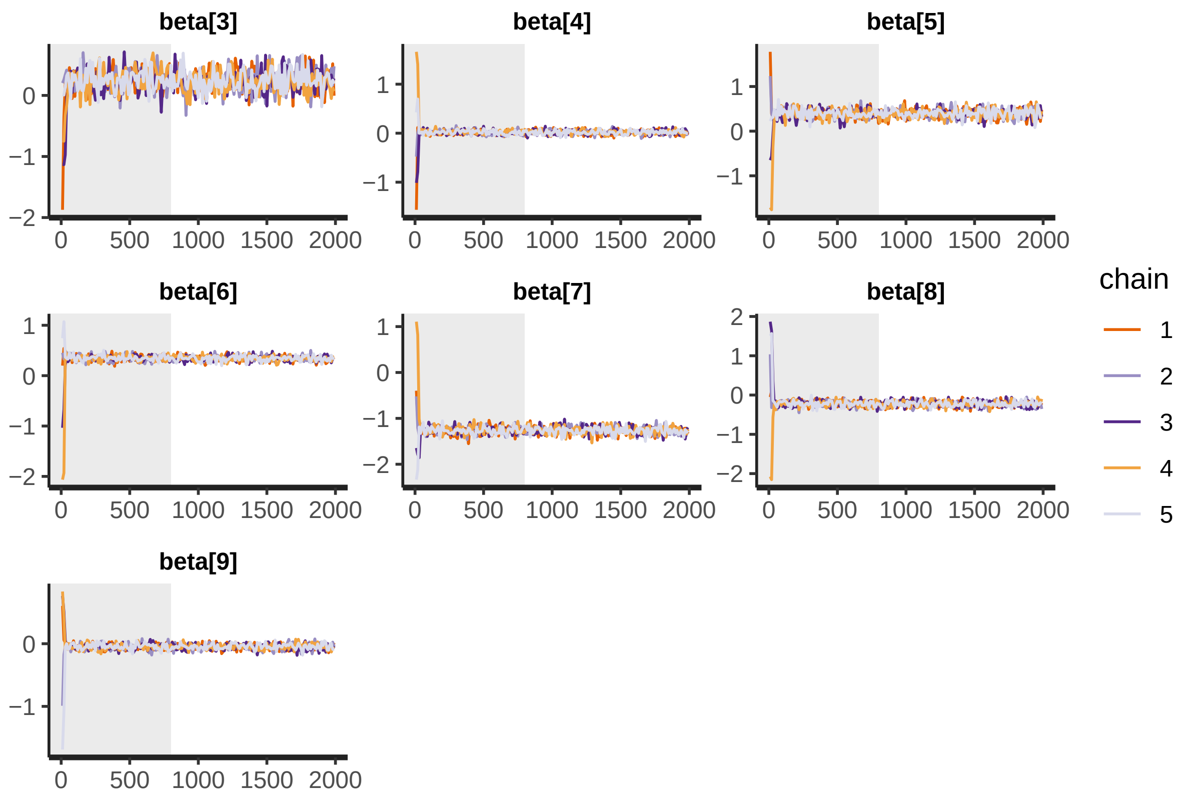
```r
## Create Stan data
datFullBayes <- list(N      = nrow(datSetCluster),
                     p      = ncol(datSetCluster) - 2,
                     died   = as.numeric(datSetCluster$Died),
                     urban  = as.numeric(datSetCluster$Urban),
                     year   = as.numeric(datSetCluster$Year),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

## Load Stan file

fileName <- "./logistic_regression_model.stan"
stanCodeFull <- readChar(fileName, file.info(fileName)$size)
cat(stanCodeFull)
```

```
## 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);
## }
```

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



Comparison between frequentist and bayesian approach

```
## Bayesian
print(resStanFull, pars = c('beta'))
```

```
## Inference for Stan model: a34018df5244a5850ca83f71127f5795.
## 5 chains, each with iter=2000; warmup=800; thin=10;
## post-warmup draws per chain=120, total post-warmup draws=600.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  0.48    0.01 0.34 -0.17  0.23  0.49  0.70  1.16   683 1.00
## beta[2]  0.30    0.00 0.06  0.18  0.26  0.31  0.35  0.42   519 1.00
## beta[3]  0.24    0.01 0.17 -0.11  0.12  0.23  0.34  0.59   572 1.00
## beta[4]  0.02    0.00 0.05 -0.07 -0.01  0.02  0.05  0.11   565 0.99
## beta[5]  0.40    0.00 0.10  0.19  0.33  0.40  0.47  0.60   687 1.00
## beta[6]  0.34    0.00 0.05  0.24  0.31  0.34  0.38  0.45   606 1.00
## beta[7] -1.26    0.00 0.08 -1.43 -1.31 -1.26 -1.20 -1.10   742 1.00
## beta[8] -0.22    0.00 0.08 -0.37 -0.27 -0.22 -0.17 -0.08   610 1.00
## beta[9] -0.05    0.00 0.05 -0.14 -0.08 -0.05 -0.02  0.04   563 1.00
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:26:29 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).
```

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

```
##                           coef [confint]       p
## (Intercept)                0.47 [-0.21, 1.14]   0.176
## as.numeric(Urban)          0.24 [-0.08, 0.56]   0.144
## as.numeric(Year)           0.31 [0.18, 0.43]   <0.001
## as.numeric(Season)         0.02 [-0.07, 0.11]   0.729
## as.numeric(Sex)            0.39 [0.19, 0.59]   <0.001
## as.numeric(Age)            0.34 [0.24, 0.45]   <0.001
## as.numeric(Education)     -1.26 [-1.42, -1.10] <0.001
## as.numeric(Occupation)    -0.22 [-0.37, -0.07]  0.004
## as.numeric(method)        -0.05 [-0.14, 0.04]   0.286
```

After this first analysis, as aforementioned, we decided to use the Bayesian approach.

Therefore, regarding the analysis that will follow, we first developed two models to evaluate the incidence of the factors on fatality of attempts:

- **Full logistic regression model** where all the parameters are included, as the one shown before;
- **Reduced logistic regression model** that only includes the parameters selected in the variable selection phase.

At the end of the analysis we have devoloped two other models to predict the age of the attempters. These two models correspond to a full model with all the parameters and a reduced one with only the most relevant parameters once again.

Prior choices

For the fist models (where we evaluated the incidence of the parameters on the fatality of attempts) we have assumed flat priors while in the last two models we have decided to assign a weakly informative prior to the standard deviation $\sigma \sim N(0, 10)$ since the mean μ was too sensible to prior choices.

Code

Data loading

First of all we load the data. Notice that some processing was done on the original data removing samples with missing entries (that resulted to constitute less than the 6 % of the dataset) and turning labels from strings into integers.

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

In this phase we are working on testing different models, therefore 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(datSetCluster$Person_ID, size = 50, replace = TRUE)

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

```
## Create Stan data
dat_red <- 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))
```

Full logistic regression model

Here we start by implementing the full logistic regression model.

```
## FULL LOGISTIC REGRESSION MODEL

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

```
## 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);
## }
```

Stan Code Running

The Stan models are run by using five chains constituted by 2000 iterations, a warmup length of 800 iterations and a thin equal to 10. Thin is a positive integer that specifies the period for saving samples; it is set by default = 1, and it is normally left to defaults. In our case though our posterior distribution takes up a lot of memory even when using a reduced dataset and we require a large numer of iteration to achieve effective sample size and therefore we decide to set it to 10 in this phase.

```
## 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
```

```
## Inference for Stan model: a34018df5244a5850ca83f71127f5795.
## 5 chains, each with iter=2000; warmup=800; thin=10;
## post-warmup draws per chain=120, total post-warmup draws=600.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1] -1.27    0.15 3.78 -9.56 -3.78 -1.11  1.35  5.51   643    1
## beta[2]  1.06    0.03 0.63 -0.06  0.63  1.04  1.50  2.29   579    1
## beta[3]  1.25    0.09 2.14 -2.95 -0.18  1.24  2.69  5.31   610    1
## beta[4]  0.06    0.02 0.43 -0.72 -0.23  0.04  0.32  0.93   573    1
## beta[5] -1.12    0.05 1.13 -3.45 -1.84 -1.05 -0.29  0.98   602    1
## beta[6]  0.10    0.03 0.64 -1.14 -0.36  0.09  0.52  1.40   605    1
## beta[7] -2.63    0.04 0.98 -4.82 -3.25 -2.61 -1.93 -0.93   577    1
## beta[8]  0.11    0.03 0.84 -1.63 -0.41  0.09  0.70  1.70   625    1
## beta[9]  0.68    0.02 0.44 -0.13  0.38  0.67  0.96  1.57   605    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:27:03 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).
```

Variable selection

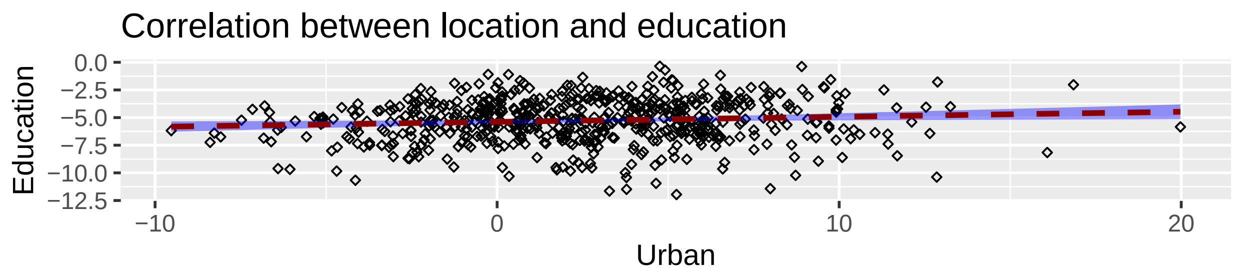
In this section we will evaluate the most influential factors and their correlation in order to select the most descriptive ones that will be used to contruct our second model (the reduced logistic regression model).

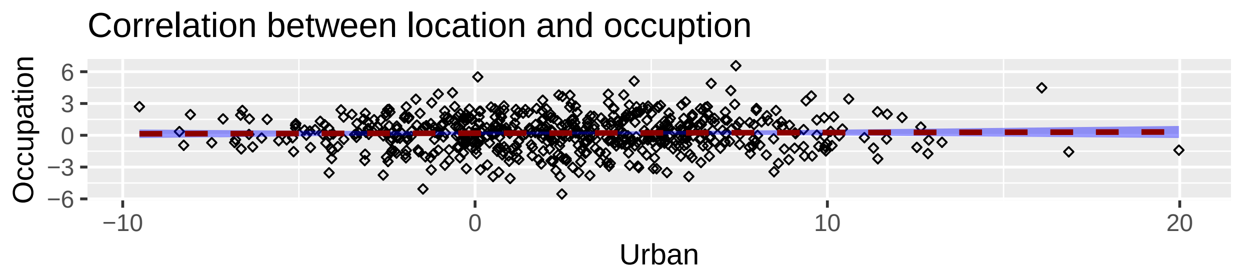
First of all we process our data:

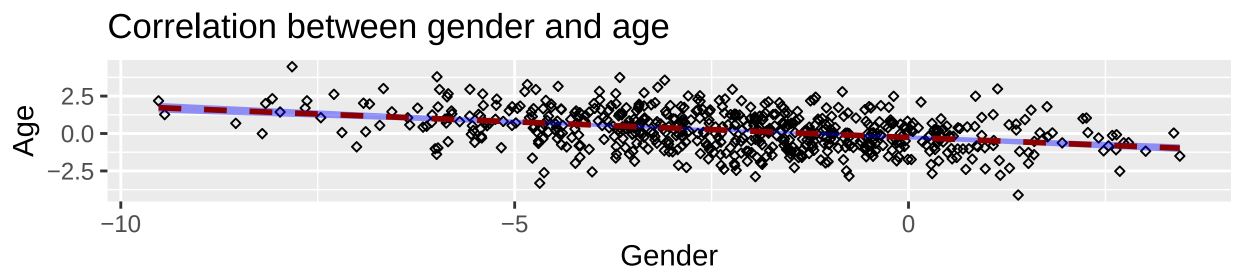
```
# Transform fitting over beta in a dataframe for the plots

beta_matrix <- zeros(length(extract(resStanFull)$beta[,1]), ncol(data_reduced) - 2)

for (i in 1:ncol(data_reduced) - 2)
  beta_matrix[,i] = beta_matrix[,i] + extract(resStanFull)$beta[,i]

beta_df <- as.data.frame(beta_matrix)
```

Now we show traceplots and generate scatter plots in order to evaluate the correlation between the parameters:

```
# Generate some scatter plots in order to see the correlations between parameters
scatter_1 <- ggplot(beta_df, aes(x=V3, y=V7)) +
                    ggtitle("Correlation between location and education") +
                    xlab("Urban") + ylab("Education") +
             geom_point(size=1, shape=23) +
             geom_smooth(method=lm, linetype="dashed", color="darkred", fill="blue")

scatter_2 <- ggplot(beta_df, aes(x=V3, y=V8)) +
                    ggtitle("Correlation between location and occuption") +
                    xlab("Urban") + ylab("Occupation") +
             geom_point(size=1, shape=23) +
             geom_smooth(method=lm, linetype="dashed", color="darkred", fill="blue")

scatter_3 <- ggplot(beta_df, aes(x=V5, y=V6)) +
                    ggtitle("Correlation between gender and age") +
                    xlab("Gender") + ylab("Age") +
             geom_point(size=1, shape=23) +
             geom_smooth(method=lm, linetype="dashed", color="darkred", fill="blue")


ggplot2.multiplot(scatter_1,scatter_2,scatter_3, cols=1)
```

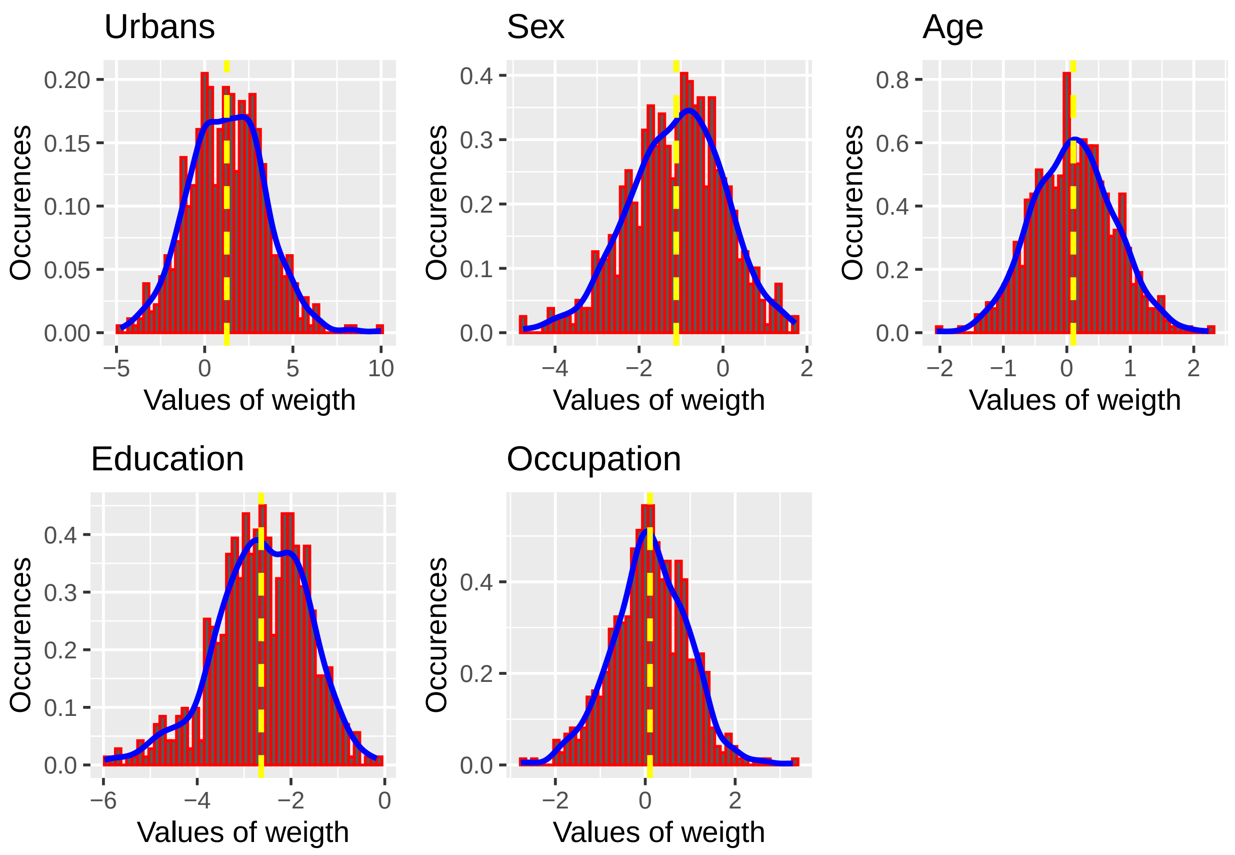






Now we overlay histogram, density and mean value of the parameters. The most interesting plots are presented; using the mean value is interesting since we can understand which are the parameters that influence more the posterior. Thus, looking at the weight of the parameters in the histograms, it is possible to suppose with enough precision which are the most informative parameters.

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

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

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

plot_4 <- qplot(extract(resStanFull)$beta[,7], geom = 'blank',
                xlab = 'Values of weigth', ylab = 'Occurences', main='Education') +
  geom_histogram(aes(y = ..density..),col = I('red'), bins = 50) +
  geom_line(aes(y = ..density..), size = 1, col = I('blue'), stat = 'density', ) +
  geom_vline(aes(xintercept=mean(extract(resStanFull)$beta[,7])), col=I('yellow'), linetype="dashed", si

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

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 urban or rural areas, then their education, occupation and partially if they are man or woman. 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.

Full logistic regression model

For completeness we report here the full model once again.

```
## FULL LOGISTIC REGRESSION MODEL

## Load Stan Model
fileNameOne <- "./logistic_regression_model.stan"
```

```r
stan_code_full <- readChar(fileNameOne, file.info(fileNameOne)$size)
cat(stan_code_full)
```

```
## 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);
## }
```

Reduced logistic regression model

We can now implement the reduced logistic regression model using the selected parameters.

```r
## REDUCED LOGISTIC REGRESSION MODEL
## Load Stan Model
fileNameOneDef <- "./logistic_regression_model_def.stan"
stan_code_simple_def <- readChar(fileNameOneDef, file.info(fileNameOneDef)$size)
cat(stan_code_simple_def)
```

```
## 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>  sex[N];
##   int<lower=0>  age[N];
##   int<lower=0>  edu[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]*sex[i]  + beta[3]*age[i] +
##                             beta[4]*edu[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);
## }
```

Stan Code Running

Now we are going to run the model on the full dataset.
First we define the Stan data to run the second (reduced) model.

```r
## Create Stan data
dat_def <- list(N        = nrow(datSetCluster),
                p        = 4,
                died     = as.numeric(datSetCluster$Died),
                sex      = as.numeric(datSetCluster$Sex),
                age      = as.numeric(datSetCluster$Age),
                 edu      = as.numeric(datSetCluster$Education))
```

We first run the full model. The settings are the same as before except that now we are using the full dataset and default value for thin.

```
## Inference for Stan model: a34018df5244a5850ca83f71127f5795.
## 5 chains, each with iter=2000; warmup=800; thin=1;
## post-warmup draws per chain=1200, total post-warmup draws=6000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  0.47    0.01 0.34 -0.19  0.24  0.48  0.70  1.14  2853    1
## beta[2]  0.31    0.00 0.06  0.18  0.26  0.31  0.35  0.43  5363    1
## beta[3]  0.24    0.00 0.17 -0.08  0.13  0.25  0.35  0.57  5294    1
## beta[4]  0.01    0.00 0.05 -0.07 -0.02  0.01  0.05  0.10  5254    1
## beta[5]  0.39    0.00 0.10  0.19  0.32  0.39  0.46  0.59  5569    1
## beta[6]  0.34    0.00 0.05  0.24  0.31  0.35  0.38  0.45  3893    1
## beta[7] -1.26    0.00 0.08 -1.43 -1.32 -1.26 -1.21 -1.10  4379    1
## beta[8] -0.22    0.00 0.08 -0.37 -0.27 -0.22 -0.17 -0.07  4955    1
## beta[9] -0.05    0.00 0.05 -0.14 -0.08 -0.05 -0.02  0.04  4464    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:34:30 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).
```

Now we run the reduced order model..

```
## Inference for Stan model: 42f8f42f99ebe8f9576f152a7e8a0f9d.
## 5 chains, each with iter=2000; warmup=800; thin=1;
## post-warmup draws per chain=1200, total post-warmup draws=6000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  0.63       0 0.23  0.19  0.48  0.63  0.79  1.07  2261    1
## beta[2]  0.45       0 0.10  0.25  0.38  0.45  0.52  0.64  3268    1
## beta[3]  0.35       0 0.05  0.25  0.31  0.35  0.39  0.45  2654    1
## beta[4] -1.23       0 0.08 -1.39 -1.28 -1.23 -1.17 -1.07  2476    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:39:22 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).
```

Convergence Analysis

In this section we are going to analyse the implemented models, both in terms of convergence (assessed using R-hat and HMC specific convergence diagnostic) and efficiency (by computing the Effective Sample Size).

R-hat

R-hat convergence diagnostic compares between- and within-chain estimates for model parameters and other univariate quantities of interest. If chains have not mixed well R-hat is larger than 1. In practical terms, it is good practice to use at least four chains and using the sample if R-hat is less than 1.05.
We can see from the result of `print(fit)` we have just displayed that all the Rhat values are equal to one for both the models and therefore we have convergence.

HMC

Here we compute convergence diagnostic specific to Hamiltonian Monte Carlo, and in particular divergences and tree depth.
The following code computes the diagnostic for the full model:

```r
## Full model HMC diagnostic

check_hmc_diagnostics(resStanFull)
```

```
##
## Divergences:

## 0 of 6000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 6000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

As we can see none of the interations ended with a divergence nor saturated the maximum tree depth.
Now we compute the diagnostic for the reduced model:

```r
## Reduced model HMC diagnostic

check_hmc_diagnostics(resStanRed)
```

```
##
## Divergences:

## 0 of 6000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 6000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

Also for the reduced model none of the iterations ended with a divergence nor saturated the maximum tree depth.

ESS

Effective sample size (ESS) measures the amount by which autocorrelation within the chains increases uncertainty in estimates.
As for the Rhat values we can directly observe the effective sample size values of the chains using the command `print(fit)`, already used. We can see that the sample size values are all sufficiently high for both model.

Posterior Predictive Checking

In order to develop a precise posterior predictive checking and a model comparison, it was decided to use the built-in Stan function `stan_glm`. This was mainly done for treatability reason. As a matter of fact, with the defined function it is easier to create different Stan models, add priors and genearate new sample from the posterior.

```r
cat('\n Summary of null model \n')
```

```
##
##  Summary of null model
```

```r
summary(model.null)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      as.numeric(Died) ~ 1
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2330
##  predictors:   1
##
## Estimates:
##               mean   sd    10%   50%   90%
## (Intercept)  -0.4    9.9 -13.0  -0.2  12.3
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.5  0.0   0.5   1.0
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.3  1.0  1271
## mean_PPD      0.0  1.0  1697
## log-posterior 0.0  1.0   917
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
cat('\n Summary of full model \n')
```

```
##
##  Summary of full model
```

```r
summary(model.full)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      as.numeric(Died) ~ as.numeric(Urban) + as.numeric(Year) + as.numeric(Sex) +
##     as.numeric(Age) + as.numeric(Education) + as.numeric(Occupation) +
##     as.numeric(method) + as.numeric(Season)
##  algorithm:    sampling
##  sample:       5000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2330
##  predictors:   9
##
```

```
## Estimates:
##                            mean   sd   10%   50%   90%
## (Intercept)               0.5    0.4  0.0   0.5   0.9
## as.numeric(Urban)         0.2    0.2  0.0   0.2   0.5
## as.numeric(Year)          0.3    0.1  0.2   0.3   0.4
## as.numeric(Sex)           0.4    0.1  0.3   0.4   0.5
## as.numeric(Age)           0.3    0.1  0.3   0.3   0.4
## as.numeric(Education)  -1.3    0.1 -1.4  -1.3  -1.2
## as.numeric(Occupation) -0.2    0.1 -0.3  -0.2  -0.1
## as.numeric(method)        0.0    0.0 -0.1   0.0   0.0
## as.numeric(Season)        0.0    0.0  0.0   0.0   0.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                        mcse Rhat n_eff
## (Intercept)            0.0  1.0  7248
## as.numeric(Urban)      0.0  1.0  7612
## as.numeric(Year)       0.0  1.0  7049
## as.numeric(Sex)        0.0  1.0  7547
## as.numeric(Age)        0.0  1.0  9222
## as.numeric(Education)  0.0  1.0  6715
## as.numeric(Occupation) 0.0  1.0  8007
## as.numeric(method)     0.0  1.0  7516
## as.numeric(Season)     0.0  1.0  7794
## mean_PPD               0.0  1.0  5205
## log-posterior          0.0  1.0  2239
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
cat('\n Summary of reduced model \n')
```

```
##
##  Summary of reduced model
```

```r
summary(model.reduced)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      as.numeric(Died) ~ as.numeric(Urban) + as.numeric(Sex) + as.numeric(Age) +
##     as.numeric(Education) + as.numeric(Occupation) + as.numeric(method)
##  algorithm:    sampling
##  sample:       5000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2330
##  predictors:   7
##
## Estimates:
##                            mean   sd   10%   50%   90%
## (Intercept)              1.0    0.3  0.6   1.0   1.4
## as.numeric(Urban)        0.2    0.2  0.0   0.2   0.4
## as.numeric(Sex)          0.4    0.1  0.3   0.4   0.5
## as.numeric(Age)          0.3    0.1  0.3   0.3   0.4
## as.numeric(Education)  -1.2    0.1 -1.4  -1.2  -1.1
## as.numeric(Occupation) -0.2    0.1 -0.3  -0.2  -0.1
## as.numeric(method)       0.0    0.0 -0.1   0.0   0.0
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                        mcse Rhat n_eff
## (Intercept)            0.0  1.0  5733
## as.numeric(Urban)      0.0  1.0  5712
## as.numeric(Sex)        0.0  1.0  5647
## as.numeric(Age)        0.0  1.0  5648
## as.numeric(Education)  0.0  1.0  4582
## as.numeric(Occupation) 0.0  1.0  5907
## as.numeric(method)     0.0  1.0  5573
## mean_PPD               0.0  1.0  4527
## log-posterior          0.0  1.0  2702
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
cat('\n Summary of normal model \n')
```

```
##
##  Summary of normal model
```

```r
summary(model.normal)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      as.numeric(Died) ~ as.numeric(Urban) + as.numeric(Year) + as.numeric(Sex) +
##     as.numeric(Age) + as.numeric(Education) + as.numeric(Occupation) +
##     as.numeric(method) + as.numeric(Season)
##  algorithm:    sampling
##  sample:       5000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2330
##  predictors:   9
##
## Estimates:
##                          mean   sd   10%   50%   90%
## (Intercept)              0.6    0.1  0.5   0.6   0.7
## as.numeric(Urban)        0.0    0.0  0.0   0.0   0.1
## as.numeric(Year)         0.1    0.0  0.0   0.1   0.1
## as.numeric(Sex)          0.1    0.0  0.0   0.1   0.1
## as.numeric(Age)          0.1    0.0  0.1   0.1   0.1
## as.numeric(Education)  -0.2    0.0 -0.3  -0.2  -0.2
## as.numeric(Occupation)  0.0    0.0 -0.1   0.0   0.0
## as.numeric(method)      0.0    0.0  0.0   0.0   0.0
## as.numeric(Season)      0.0    0.0  0.0   0.0   0.0
## sigma                   0.4    0.0  0.4   0.4   0.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                        mcse Rhat n_eff
## (Intercept)            0.0  1.0  8577
## as.numeric(Urban)      0.0  1.0  8687
## as.numeric(Year)       0.0  1.0  8087
## as.numeric(Sex)        0.0  1.0  8191
## as.numeric(Age)        0.0  1.0  7260
## as.numeric(Education)  0.0  1.0  7725
## as.numeric(Occupation) 0.0  1.0  7866
## as.numeric(method)     0.0  1.0  8401
## as.numeric(Season)     0.0  1.0  8806
## sigma                  0.0  1.0  7347
## mean_PPD               0.0  1.0  5848
## log-posterior          0.0  1.0  2492
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
cat('\n Summary of Possion model \n')
```

```
##
##  Summary of Possion model
```

```r
summary(model.poisson)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      as.numeric(Died) ~ as.numeric(Urban) + as.numeric(Year) + as.numeric(Sex) +
##     as.numeric(Age) + as.numeric(Education) + as.numeric(Occupation) +
##     as.numeric(method) + as.numeric(Season)
##  algorithm:    sampling
##  sample:       5000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2330
##  predictors:   9
##
## Estimates:
##                          mean   sd   10%   50%   90%
## (Intercept)            -0.8    0.2 -1.1  -0.8  -0.5
## as.numeric(Urban)       0.1    0.1 -0.1   0.1   0.2
## as.numeric(Year)        0.1    0.0  0.1   0.1   0.1
## as.numeric(Sex)         0.1    0.1  0.1   0.1   0.2
## as.numeric(Age)         0.2    0.0  0.1   0.2   0.2
## as.numeric(Education)  -0.4    0.0 -0.5  -0.4  -0.4
## as.numeric(Occupation) -0.1    0.0 -0.1  -0.1   0.0
## as.numeric(method)      0.0    0.0 -0.1   0.0   0.0
## as.numeric(Season)      0.0    0.0  0.0   0.0   0.0
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                        mcse Rhat n_eff
## (Intercept)            0.0  1.0  5223
## as.numeric(Urban)      0.0  1.0  5046
## as.numeric(Year)       0.0  1.0  5353
## as.numeric(Sex)        0.0  1.0  5291
## as.numeric(Age)        0.0  1.0  5469
## as.numeric(Education)  0.0  1.0  5576
## as.numeric(Occupation) 0.0  1.0  5109
## as.numeric(method)     0.0  1.0  4305
## as.numeric(Season)     0.0  1.0  4743
## mean_PPD               0.0  1.0  4981
## log-posterior          0.0  1.0  2365
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

Model Comparison

In this section the implemented models are compared using leave-one-out cross-validation.

```
loo.null    <- loo(model.null, cores = 4)
```

```
## Warning: Found 2330 observations with a pareto_k > 0.7. With this many problematic observations we r
```

```
loo.full    <- loo(model.full, cores = 4)
loo.reduced <- loo(model.reduced, cores = 4)
loo.normal   <- loo(model.normal, cores = 4)
loo.possion <- loo(model.poisson, cores = 4)

compar_models <- loo_compare(loo.null, loo.full, loo.reduced, loo.normal, loo.possion)
compar_models
```

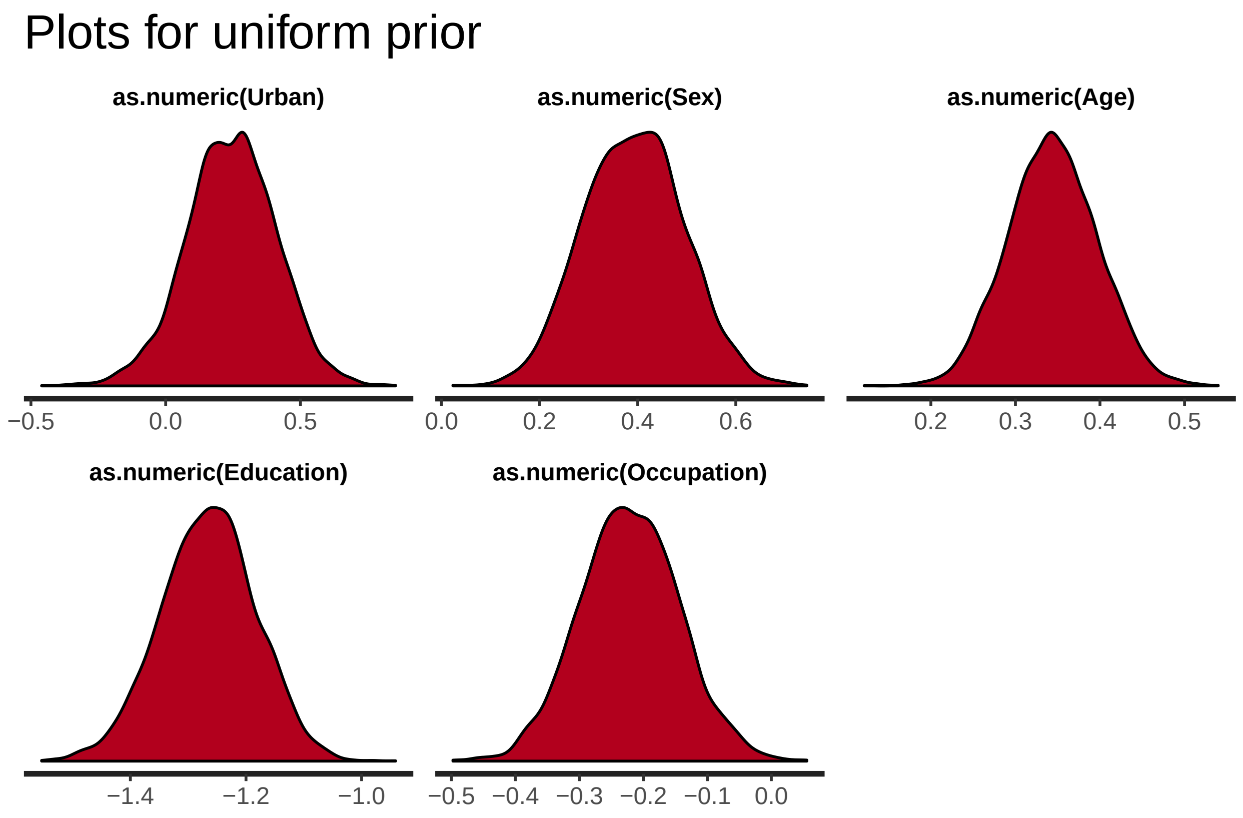
```
##                 elpd_diff se_diff
## model.full         0.0       0.0
## model.reduced    -10.0       4.9
## model.normal     -56.8       3.1
## model.poisson   -623.6      26.1
## model.null    -64390.0      53.2
```

As it is possible to see from the analysis done above, the model that performs better is the one with all the parameters and in which the family distribution is the binomial one. Regarding the result, at the beginning we though that the model could suffer for overfitting using all the parameters. However, the analysis performed, showed us that using all the parameters it is possible to obtain the best fitting.

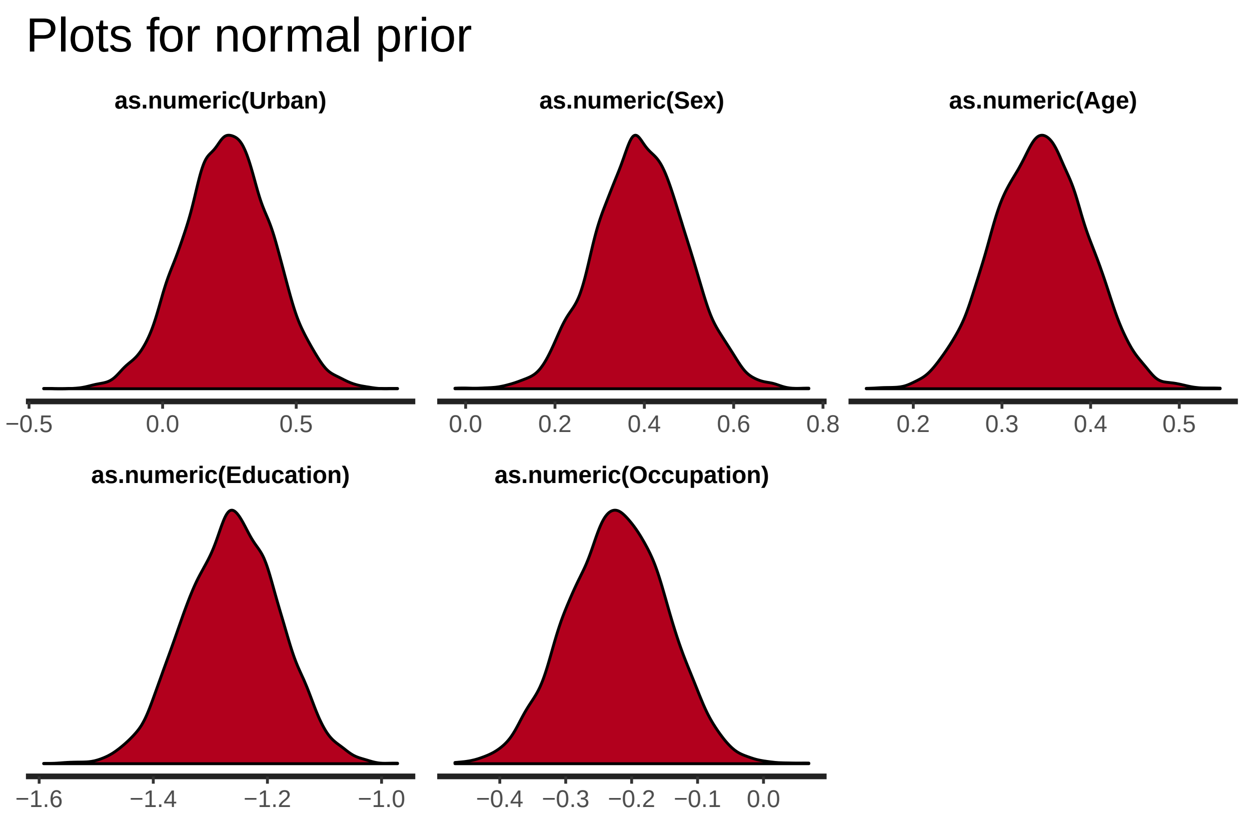
Prior sensitivity analysis

At this point, it becomes useful to develop a sensitivity analysis over the prior using the full model, i.e. the one that performed better, in order to understand what are the prior that allow to improve the posterior distribution without shifting it from the mean value.

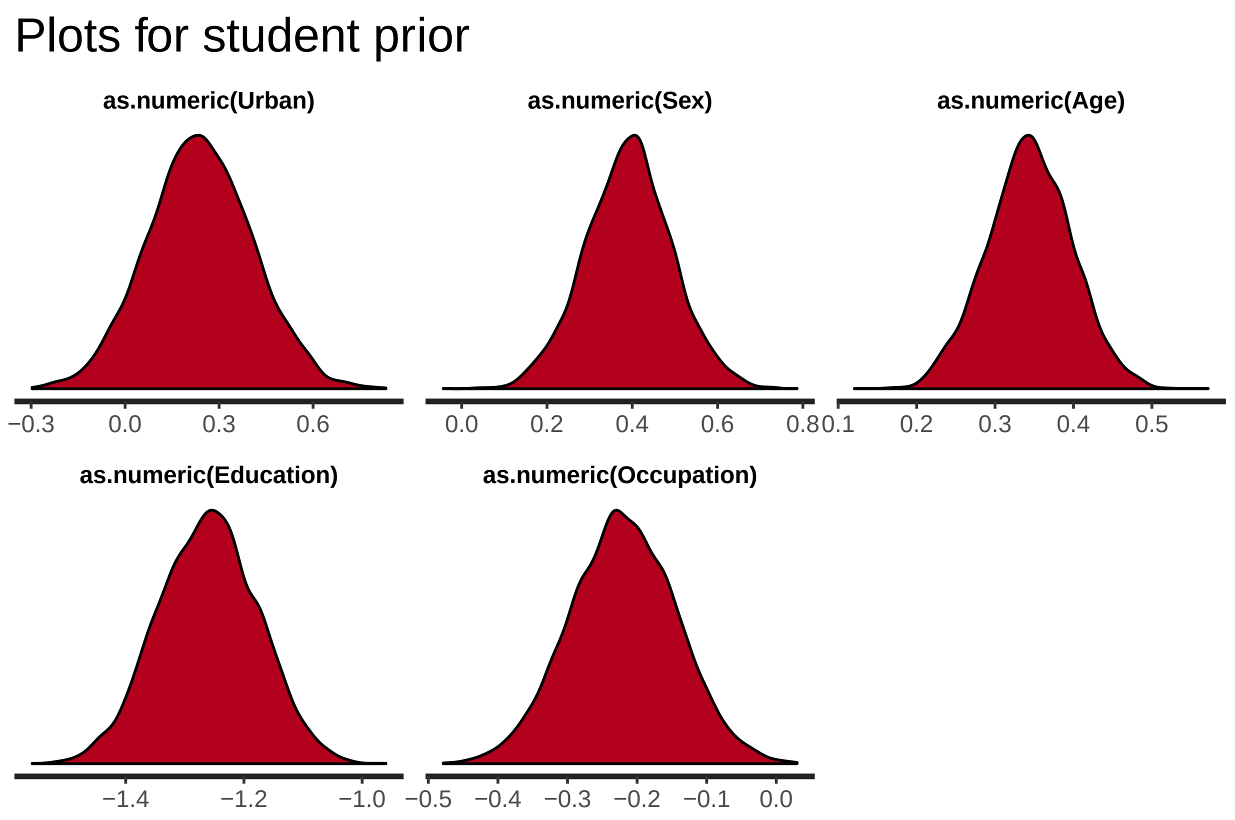
```r
# Density plots
dens.uniform <- stan_dens(model.full.uniform, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nume
dens.uniform + ggtitle('Plots for uniform prior')
```



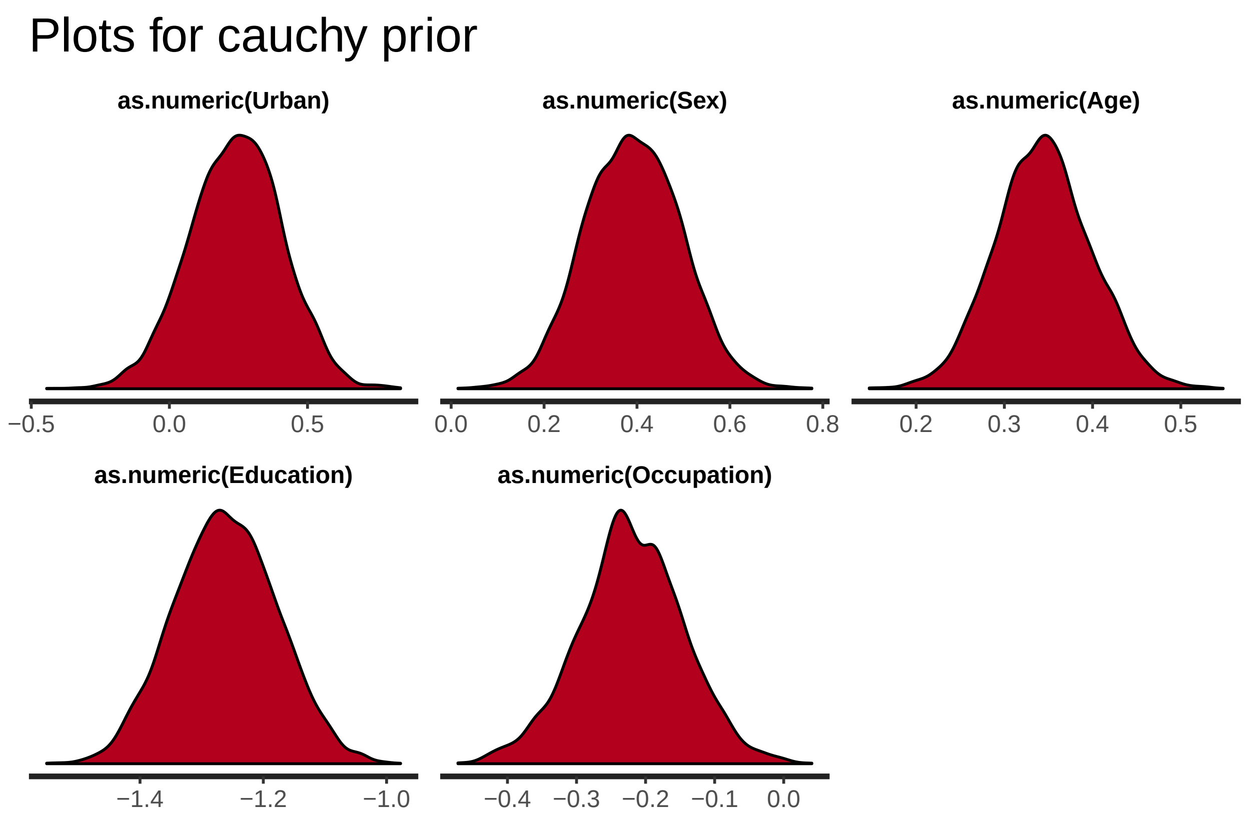
```r
dens.normal <- stan_dens(model.full.normal, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.numeri
dens.normal + ggtitle('Plots for normal prior')
```



```
dens.student <- stan_dens(model.full.student, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nume
dens.student + ggtitle('Plots for student prior')
```

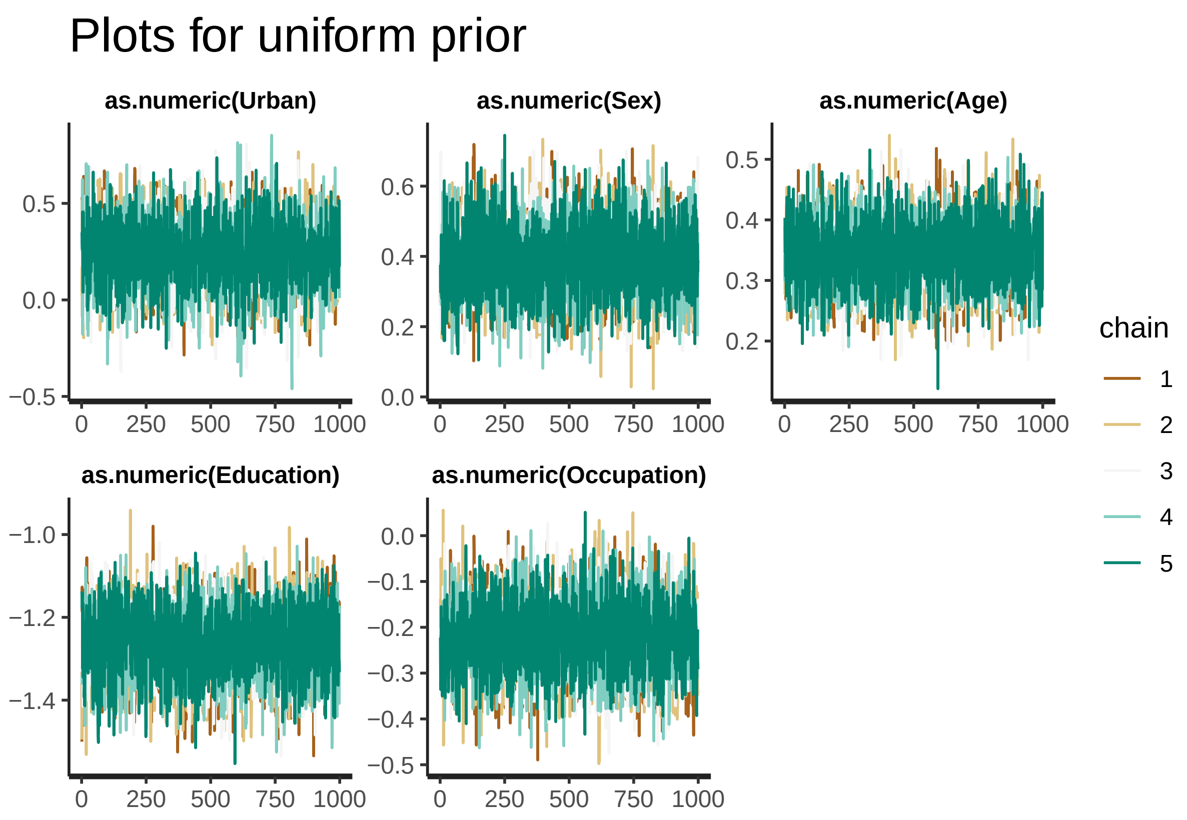


```
dens.cauchy <- stan_dens(model.full.cauchy, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.numeri
dens.cauchy + ggtitle('Plots for cauchy prior')
```



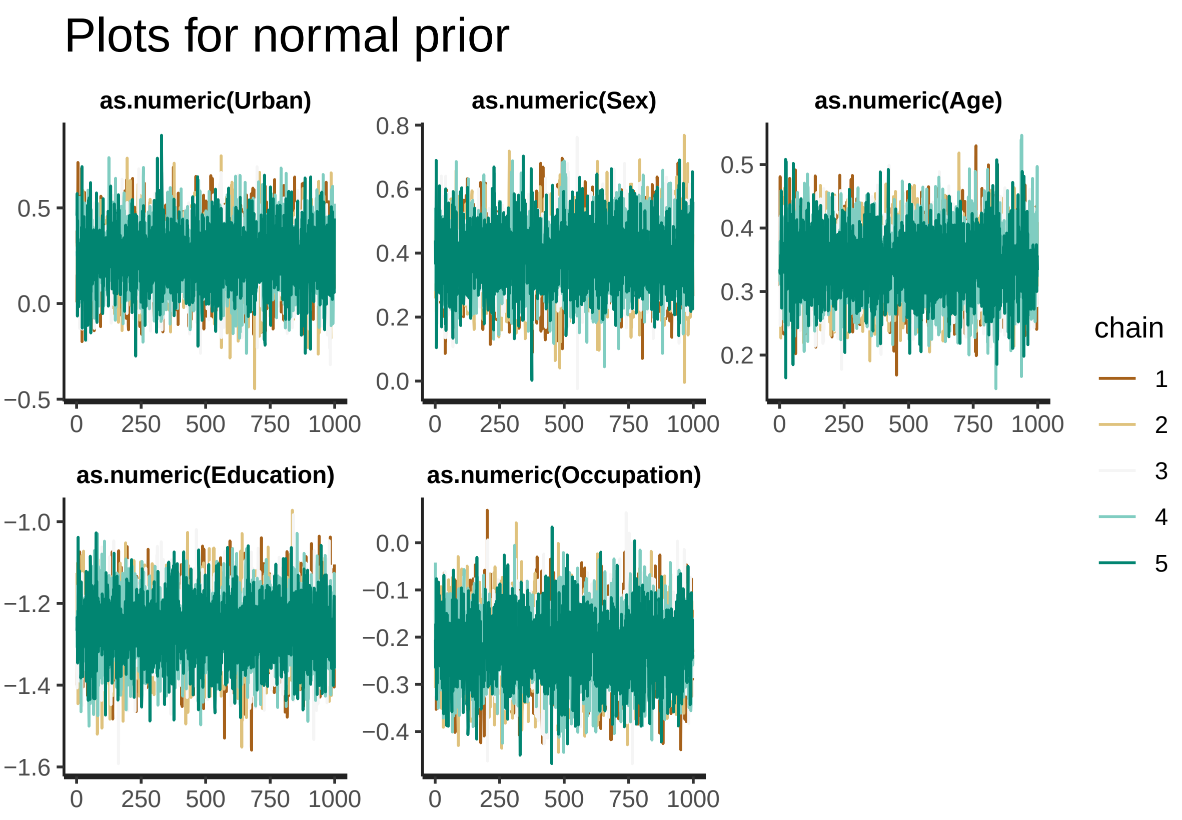
```
# Trace plots

trace.uniform <- stan_trace(model.full.uniform, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nu
trace.uniform + scale_color_brewer(type = 'div') + ggtitle('Plots for uniform prior')
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```



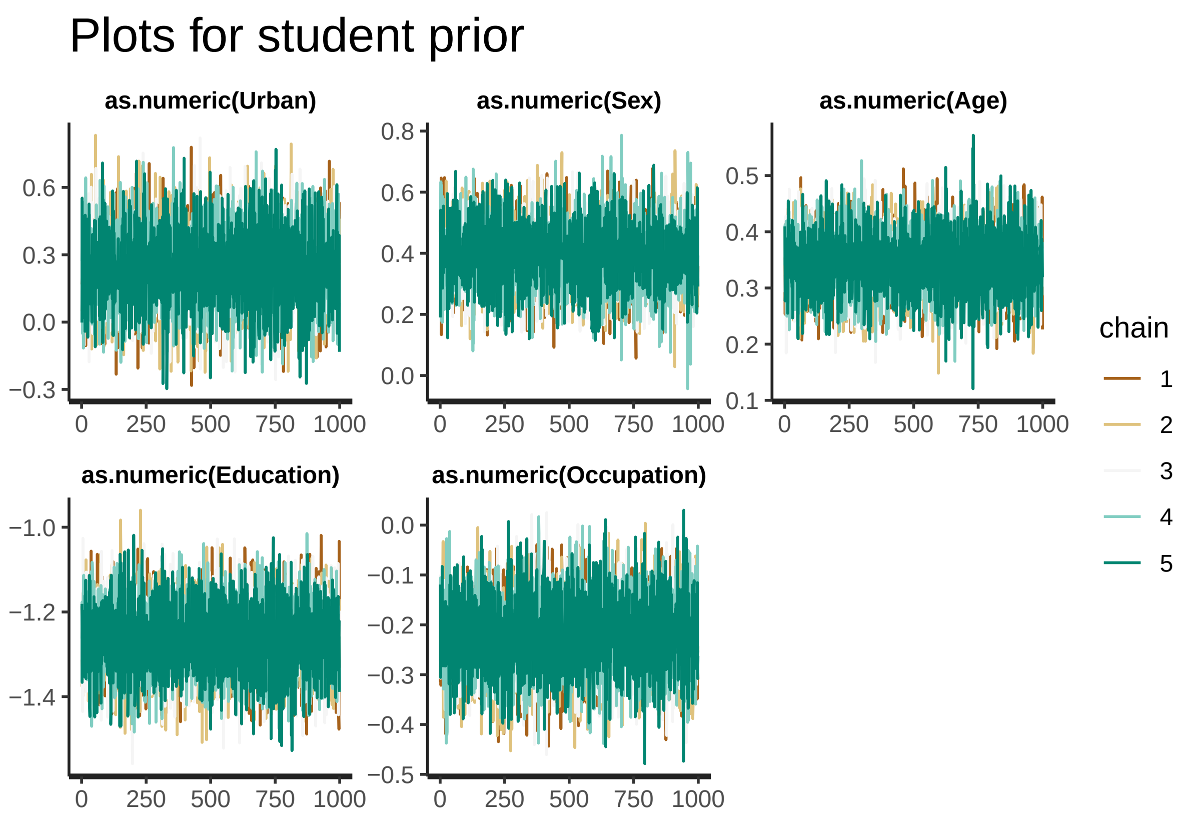
```
trace.normal <- stan_trace(model.full.normal, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nume
trace.normal + scale_color_brewer(type = 'div') + ggtitle('Plots for normal prior')
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```



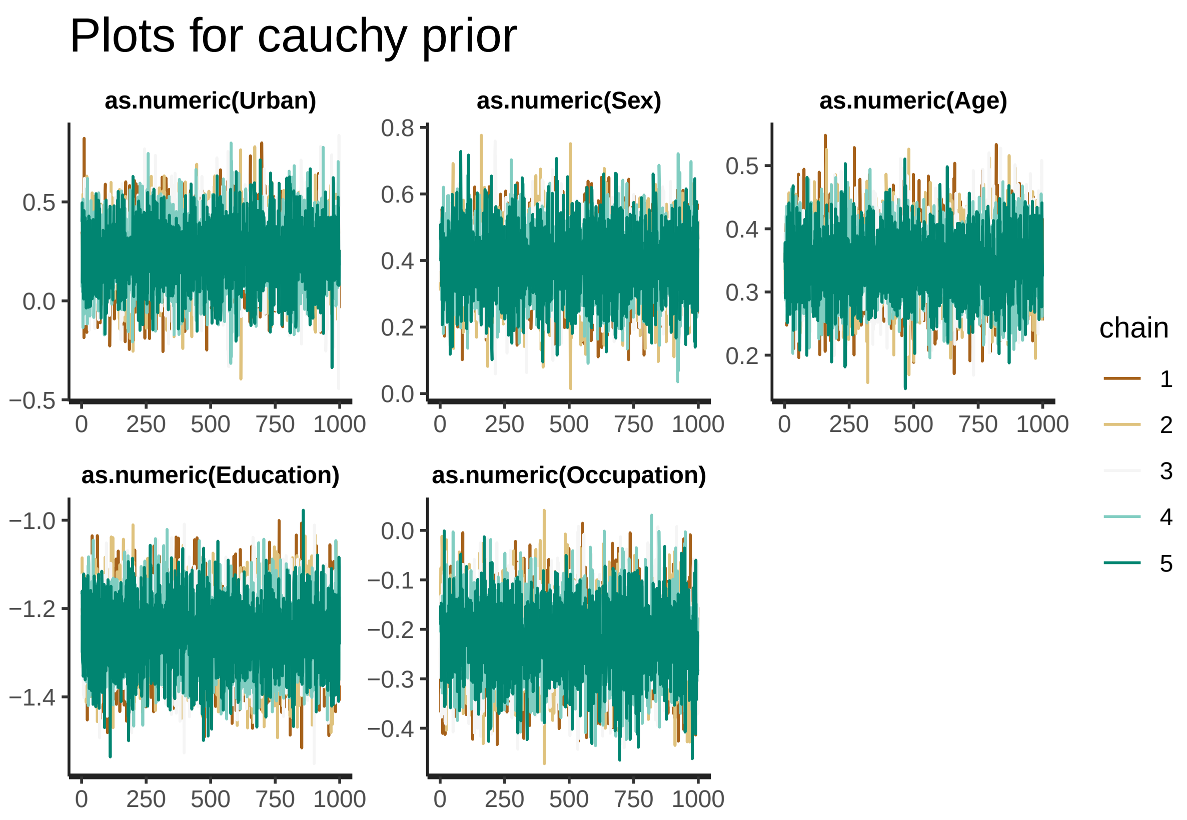
```
trace.student <- stan_trace(model.full.student, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nu
trace.student + scale_color_brewer(type = 'div') + ggtitle('Plots for student prior')
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```



```
trace.cauchy <- stan_trace(model.full.cauchy, pars = c('as.numeric(Urban)', 'as.numeric(Sex)', 'as.nume
trace.cauchy + scale_color_brewer(type = 'div') + ggtitle('Plots for cauchy prior')
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

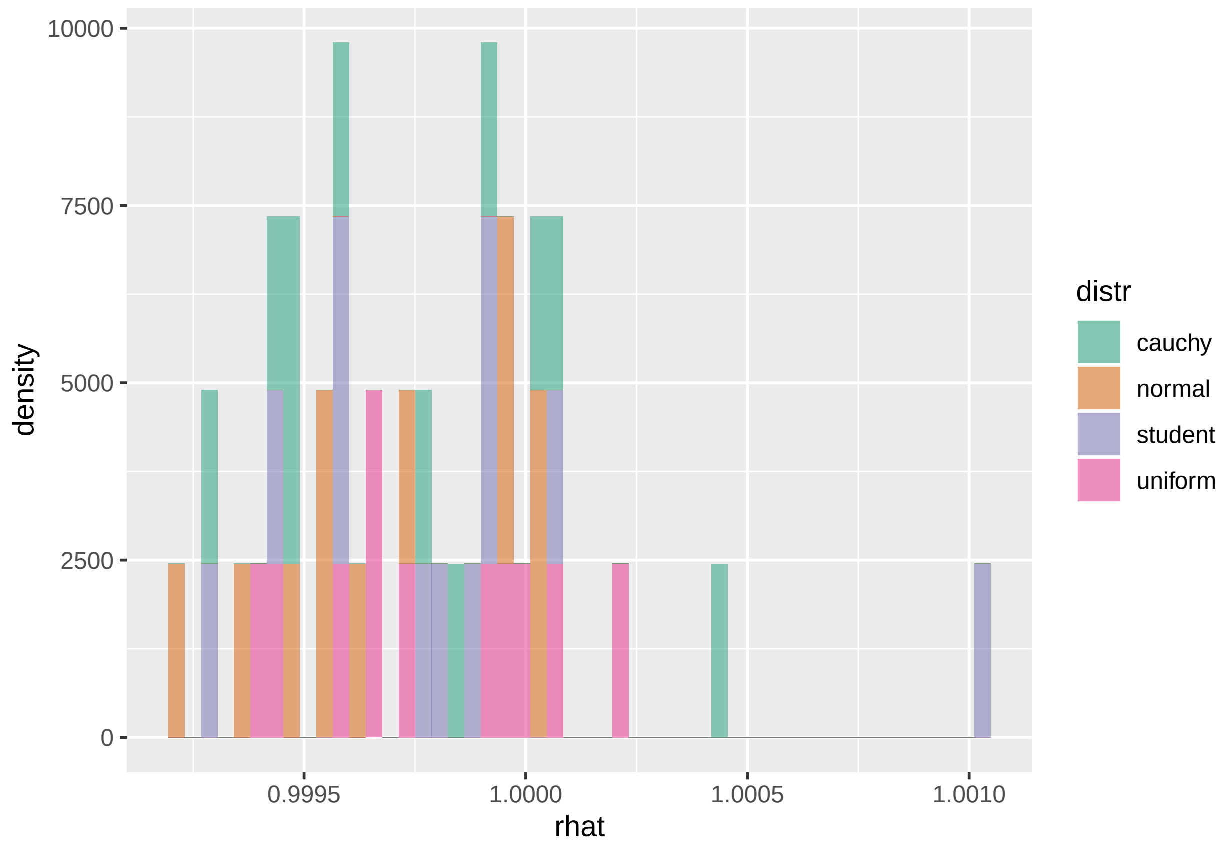


Observing the distribution above, it is clear that, even if the prior changes, the posterior distribution will remain almost the same. As a matter of fact, there are only small differences among the analysed parameters for all the models. This can be seen looking at both the plots of the distributions and to the trace plots of the chains. Moreover, all the models perform in a good way, giving convergence for all the cases.

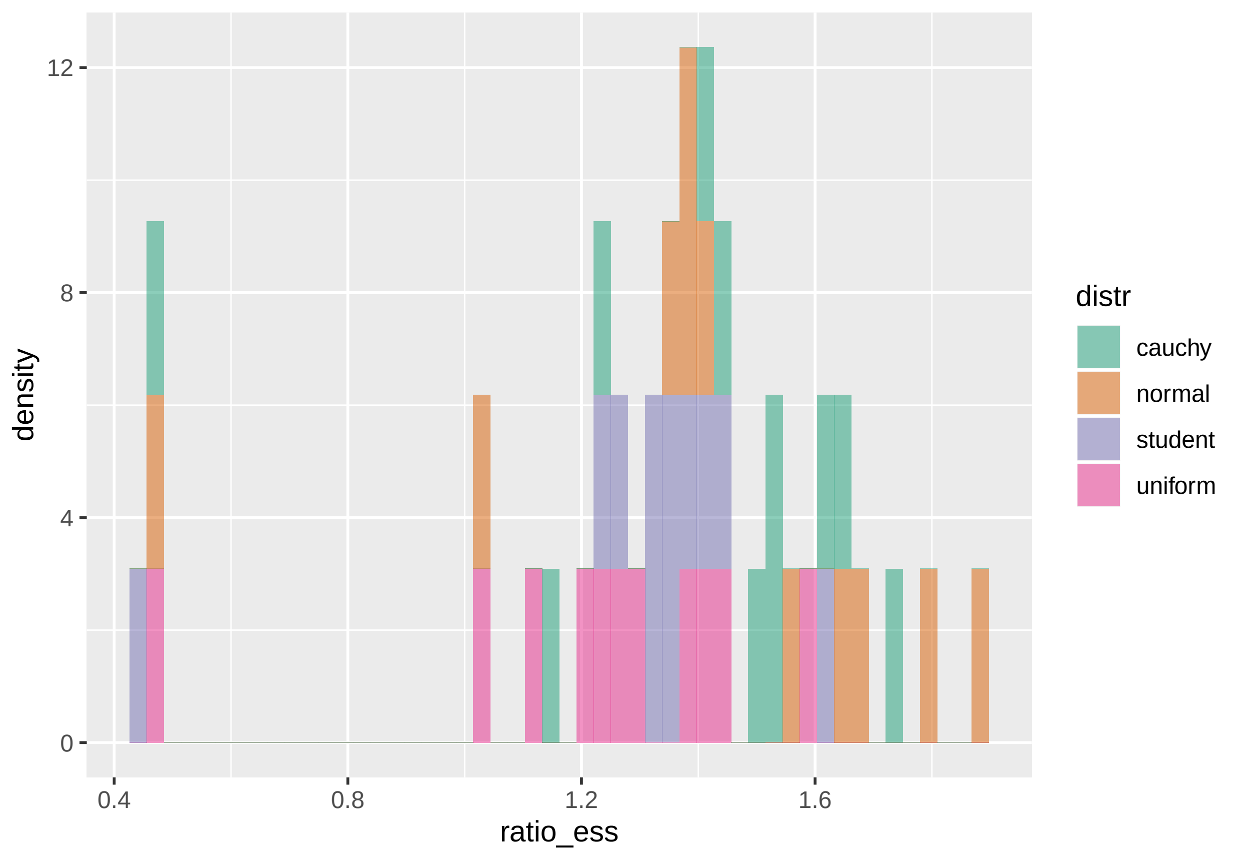
Bolow, the plots for Rhat and ESS will be proposed. As said above, the convergence is achieved, in fact the Rhat values are all below 1.05, that is the limit for the convergence. The ratio for the ESS is close to 1 too.

```
# Rhat histogram plots

rhat.uniform <- stan_rhat(model.full.uniform, bins = 50)
rhat.normal <- stan_rhat(model.full.normal, bins = 50)
rhat.student <- stan_rhat(model.full.student, bins = 50)
rhat.cauchy <- stan_rhat(model.full.cauchy, bins = 50)

uniform <- data.frame(rhat = rhat.uniform$data$stat)
normal <- data.frame(rhat = rhat.normal$data$stat)
student <- data.frame(rhat = rhat.student$data$stat)
cauchy <- data.frame(rhat = rhat.cauchy$data$stat)

uniform$distr <-  'uniform'
normal$distr  <- 'normal'
student$distr <-  'student'
cauchy$distr  <- 'cauchy'


distrLength <- rbind(uniform, normal, student, cauchy)
```

```r
ggplot(distrLength, aes(rhat, fill = distr)) + geom_histogram(alpha = 0.5, aes(y = ..density..), bins =
```



```r
# Ess histogram plots

ess.uniform <- stan_ess(model.full.uniform, bins = 50)
ess.normal <- stan_ess(model.full.normal, bins = 50)
ess.student <- stan_ess(model.full.student, bins = 50)
ess.cauchy <- stan_ess(model.full.cauchy, bins = 50)

uniform <- data.frame(ratio_ess = ess.uniform$data$stat)
normal <- data.frame(ratio_ess = ess.normal$data$stat)
student <- data.frame(ratio_ess = ess.student$data$stat)
cauchy <- data.frame(ratio_ess = ess.cauchy$data$stat)

uniform$distr <-  'uniform'
normal$distr  <- 'normal'
student$distr <-  'student'
cauchy$distr  <- 'cauchy'


distrLength <- rbind(uniform, normal, student, cauchy)

ggplot(distrLength, aes(ratio_ess, fill = distr)) + geom_histogram(alpha = 0.5, aes(y = ..density..), b
```



```r
# Leave-one-out cross-validation over the models with different priors

loo.uniform    <- loo(model.full.uniform, cores = 4)
loo.normal    <- loo(model.full.normal, cores = 4)
loo.student <- loo(model.full.student, cores = 4)
loo.cauchy   <- loo(model.full.cauchy, cores = 4)
compar_models_prior <- loo_compare(loo.uniform, loo.normal, loo.student, loo.cauchy)
compar_models_prior
```

```
##                      elpd_diff se_diff
## model.full.normal   0.0       0.0
## model.full.student -0.2       0.0
## model.full.uniform -0.3       0.1
## model.full.cauchy  -0.3       0.0
```

Observing the values of the relative elpd among the model tested in this section, the one that performs better is the one with a cauchy distribution as prior.

Age prediction

```r
data_4_fit <- list(N = nrow(datSet),
                   p = 6,
                   age = as.numeric(datSet$Age),
                   died = as.numeric(datSet$Died),
                   sex = as.numeric(datSet$Sex),
                   job = as.numeric(datSet$Occupation),
                   urban = as.numeric(datSet$Urban),
                   edu = as.numeric(datSet$Education))


fileName <- "./stan_model_with_prior.stan"
stan_code <- readChar(fileName, file.info(fileName)$size)
cat(stan_code)
```

```
## //
## // This Stan program defines a simple model, with a
## // vector of values 'y' modeled as normally distributed
## // with mean 'mu' and standard deviation 'sigma'.
## //
## // Learn more about model development with Stan at:
## //
## //    http://mc-stan.org/users/interfaces/rstan.html
## //    https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
## //
##
##
## data {
##   // Define variables in data
##   // Number of observations (an integer)
##   int<lower=0> N;
##
##   // Number of parameters
##   int<lower=0> p;
##
##   // Variables
##   real<lower=0> age[N];
##   int<lower=0>  died[N];
##   int<lower=0>  sex[N];
##   int<lower=0>  job[N];
##   int<lower=0>  urban[N];
##   int<lower=0>  edu[N];
## }
##
## parameters {
##   // A-priori mean for prediction
##   real mu_p;
##   // Define parameters to estimate
##   real beta[p];
##
##   // standard deviation (a positve real number)
##   real<lower=0> sigma;
## }
##
## transformed parameters  {
##   // Mean
##   real mu[N];
##   for (i in 1:N) {
##     mu[i] = beta[1] + beta[2]*died[i] + beta[3]*sex[i] + beta[4]*job[i] +
##             beta[5]*urban[i] + beta[6]*edu[i];
##   }
## }
```

```
##
## model {
##   // Weakly informative prior
##   mu_p ~ normal(60, 10);
##   sigma ~ normal(0,10);
##
##   // Likelihood part of the Bayesian inference
##   age ~ normal(mu, sigma);
##
## }
##
## generated quantities{
##   real predct_age;
##   vector[N] log_lik;
##
##   predct_age = normal_rng(mu_p, sigma);
##   for (i in 1:N)
##     log_lik[i] = normal_lpdf(age[i] | mu[i], sigma);
##
## }
```

```
# Run Stan
fitStan <- stan(model_code = stan_code,
                data = data_4_fit,
                chains = 5,
                iter = 2000,
                warmup = 800,
                thin = 10,
                refresh = 0,
                seed = 12345,
                control = list(adapt_delta = 0.95))
```

```
## 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
```

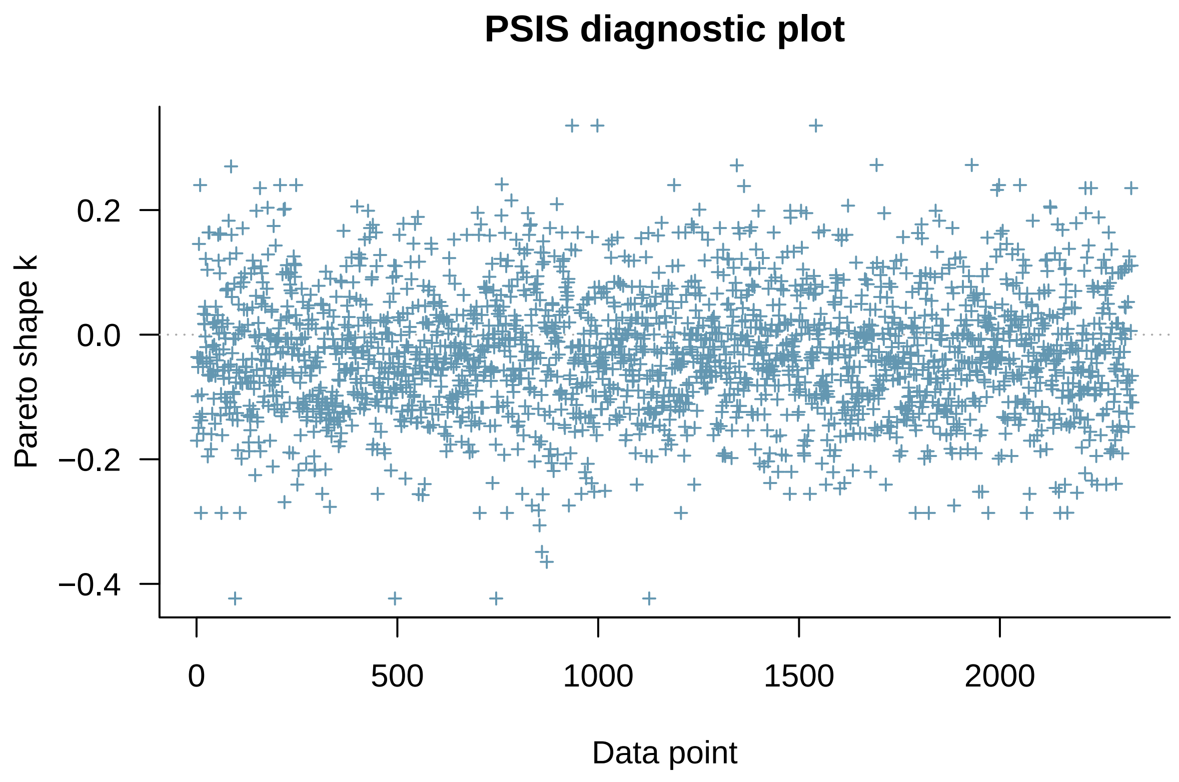
```
print(fitStan, pars = c('beta', 'sigma'))
```

```
## Inference for Stan model: 27c2703215c82db517f6e77613d1aace.
## 5 chains, each with iter=2000; warmup=800; thin=10;
## post-warmup draws per chain=120, total post-warmup draws=600.
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta[1]  67.81    0.05 1.14  65.59  67.04  67.85  68.56  69.95   597    1
## beta[2]   5.74    0.03 0.73   4.33   5.26   5.72   6.20   7.14   531    1
## beta[3]   1.85    0.03 0.63   0.70   1.41   1.85   2.27   3.12   551    1
## beta[4]  -0.38    0.02 0.48  -1.35  -0.69  -0.38  -0.08   0.54   683    1
## beta[5]  -1.03    0.04 1.04  -3.08  -1.73  -1.08  -0.30   1.06   599    1
## beta[6] -13.83    0.02 0.47 -14.71 -14.15 -13.84 -13.54 -12.89   547    1
## sigma    14.81    0.01 0.22  14.38  14.66  14.81  14.96  15.26   569    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:52:36 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).
```

```
log_like_model <- extract_log_lik(fitStan, merge_chains = FALSE)
r_eff <- relative_eff(exp(log_like_model))
loo_mod <- loo(log_like_model, r_eff = r_eff)
print(loo_mod)
```

```
##
## Computed from 600 by 2330 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo  -9591.5 34.3
## p_loo         7.6  0.3
## looic     19182.9 68.7
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_mod)
```



```
elpd_loo_mod <- loo_mod$estimates[1]
elpd_loo_mod
```

```
## [1] -9591.461
```

```r
data_4_fit_complete <- list(N = nrow(datSet),
                   p = 10,
                   age = as.numeric(datSet$Age),
                   died = as.numeric(datSet$Died),
                   hosp = as.numeric(datSet$Hospitalised),
                   year = as.numeric(datSet$Year),
                   sex = as.numeric(datSet$Sex),
                   job = as.numeric(datSet$Occupation),
                   urban = as.numeric(datSet$Urban),
                   edu = as.numeric(datSet$Education),
                   method = as.numeric(datSet$method),
                   season = as.numeric(datSet$Season))


fileName <- "./stan_model_prior_all_params.stan"
stan_code_complete <- readChar(fileName, file.info(fileName)$size)
cat(stan_code_complete)
```

```
## // This Stan program defines a simple model, with a
## // vector of values 'y' modeled as normally distributed
## // with mean 'mu' and standard deviation 'sigma'.
## //
## // Learn more about model development with Stan at:
## //
## //    http://mc-stan.org/users/interfaces/rstan.html
## //    https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
## //
##
##
## data {
##   // Define variables in data
##   // Number of observations (an integer)
##   int<lower=0> N;
##
##   // Number of parameters
##   int<lower=0> p;
##
##   // Variables
##   real<lower=0> age[N];
##   int<lower=0>  died[N];
##   int<lower=0>  hosp[N];
##   int<lower=0>  year[N];
##   int<lower=0>  sex[N];
##   int<lower=0>  job[N];
##   int<lower=0>  urban[N];
##   int<lower=0>  edu[N];
##   int<lower=0>  method[N];
##   int<lower=0>  season[N];
## }
##
## parameters {
##   // Define parameters to estimate
##   real beta[p];
##
##   // standard deviation (a positve real number)
##   real<lower=0> sigma;
## }
##
## transformed parameters  {
##   // Mean
##   real mu[N];
##   for (i in 1:N) {
##     mu[i] = beta[1] + beta[2]*died[i] + beta[3]*hosp[i] + beta[4]*year[i] +
##             beta[5]*sex[i] + beta[6]*job[i] + beta[7]*urban[i] + beta[8]*edu[i] +
##             beta[9]*method[i] + beta[10]*season[i];
##   }
## }
##
## model {
##   // Weakly informative prior
##   //mu ~ normal(0.5, 10);
##   sigma ~ normal(0,10);
##
##   // Likelihood part of the Bayesian inference
##   age ~ normal(mu, sigma);
##
## }
##
## generated quantities{
##   vector[N] log_lik;
##
##   for (i in 1:N)
##     log_lik[i] = normal_lpdf(age[i] | mu[i], sigma);
##
## }
```

```r
# Run Stan
fitStan_complete <- stan(model_code = stan_code_complete,
                data = data_4_fit_complete,
                chains = 5,
                iter = 2000,
                warmup = 800,
                thin = 10,
                refresh = 0,
                seed = 12345,
                control = list(adapt_delta = 0.95))
```

```
## 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
print(fitStan_complete, pars = c('beta', 'sigma'))
```

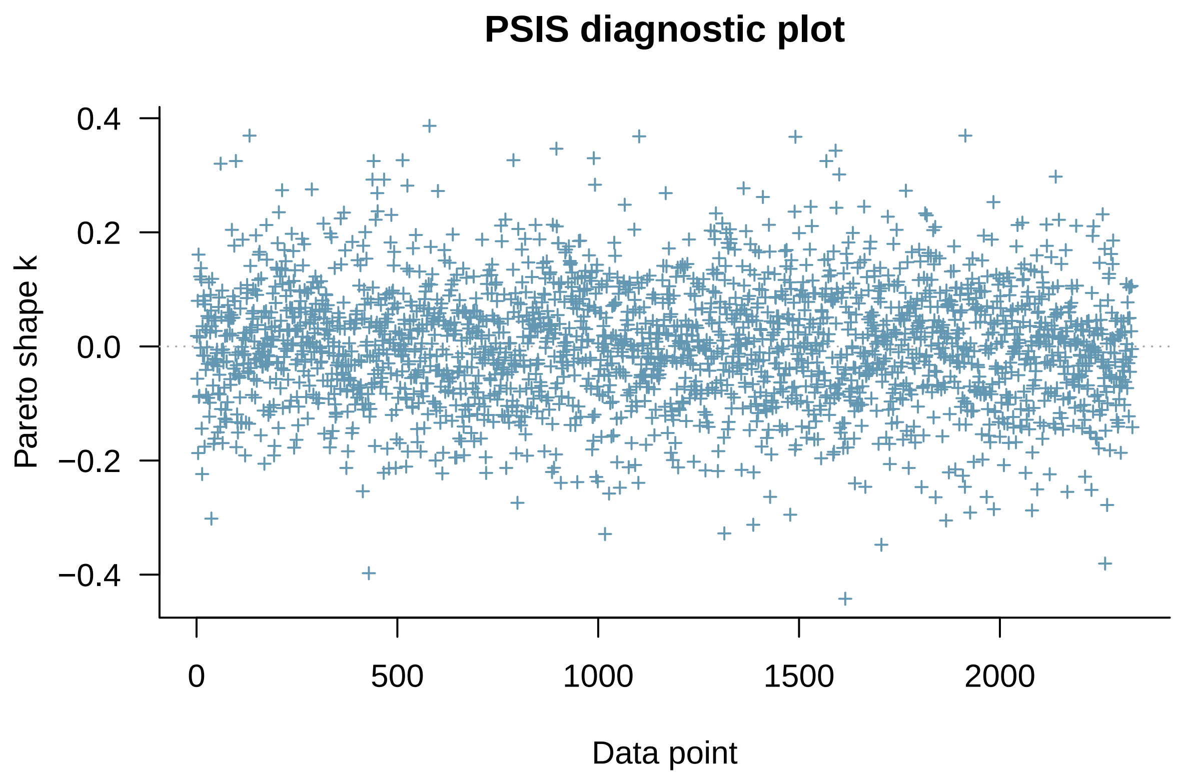
```
## Inference for Stan model: 09e3182dff98db3733ab71c0903a7bef.
## 5 chains, each with iter=2000; warmup=800; thin=10;
## post-warmup draws per chain=120, total post-warmup draws=600.
```

```
##
##            mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta[1]   78.89    0.07 1.98  74.93  77.64  78.90  80.32  82.52   703    1
## beta[2]    3.24    0.04 1.05   1.29   2.51   3.17   3.91   5.51   704    1
## beta[3]   -2.93    0.04 1.11  -5.13  -3.71  -2.98  -2.24  -0.48   640    1
## beta[4]    0.22    0.02 0.39  -0.51  -0.08   0.23   0.47   1.01   553    1
## beta[5]    1.27    0.03 0.63  -0.01   0.84   1.27   1.70   2.55   575    1
## beta[6]   -0.97    0.02 0.45  -1.83  -1.24  -0.96  -0.69  -0.08   516    1
## beta[7]   -0.71    0.04 1.03  -2.73  -1.35  -0.71  -0.06   1.42   632    1
## beta[8]  -13.41    0.02 0.42 -14.22 -13.69 -13.41 -13.13 -12.61   693    1
## beta[9]   -2.07    0.01 0.29  -2.66  -2.27  -2.06  -1.88  -1.50   575    1
## beta[10]  -0.55    0.01 0.27  -1.08  -0.73  -0.56  -0.35  -0.03   600    1
## sigma     14.63    0.01 0.22  14.23  14.48  14.62  14.77  15.03   550    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec 08 00:58:05 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).
```

```
log_like_model_compl <- extract_log_lik(fitStan_complete, merge_chains = FALSE)
r_eff <- relative_eff(exp(log_like_model_compl))
loo_mod_comp <- loo(log_like_model_compl, r_eff = r_eff)
print(loo_mod_comp)
```

```
##
## Computed from 600 by 2330 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo  -9564.3 34.6
## p_loo        11.4  0.4
## looic     19128.6 69.3
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_mod_comp)
```



```
elpd_loo_mod_comp <- loo_mod_comp$estimates[1]
elpd_loo_mod_comp
```

```
## [1] -9564.322
```

Conclusions

Problems encountered

Potential improvements