Bayesian Modelling Project

Frigerio Edoardo - Pella Claudio

Goal and Data Selection

For this project we used a dataset, from Kaggle, that contained data from houses of King's County. Our goal was to use JAGS and bayesian results to compute a model able to predict the price of a house given some parameters. We imagined to create a model as usable as possible: from the exploratory analysis we realized that most of the variables contained in the dataset where either not worth keeping or too difficult to come by and therefore we dropped them. Also, the dataset contained several thousands of lines, with 70 different zipcodes: we focused on the **city of Seattle**, excluding luxury houses.

So, our covariates: * **Price** - price of the house, re-scaled with logarithm base 10. * **Zipcode** - 15 factors, the zipcodes on the city. * **Waterfront** - dummy variable, if the house has view on the lake (1) or not (0). * **Sqft_living** - size of the house, re-scacled in square meters.

Creation of the Model

First, we divided the dataset in 2 subsets: **train** and **test**. The former will be used for training, the latter for validation, with a proportion of 90%-10%.

```
################################################################
################################################################

ID <- sample(nrow(home), nrow(home)*0.1)
test <- home[ID,]
train <- home[-ID,]

train$zipcode <- as.factor(data.matrix(train)[,4])
train$price <- log(train[,1],10)

test$zipcode <- data.matrix(test)[,4]
test$price <- log(test[,1],10)

train <- as.data.frame(train)
test <- as.data.frame(test)
```

We used a Hierarchical model. We selected non-informative priors since we had no information about the parameters. First level:

$$Y_{1,\dots,n} \sim N(\mu(i), \tau)$$

so the price is a normal with average μ and variance τ

$$\mu = \beta * X + u[zipcode] + \epsilon$$

where β is the vector of parameters of the linear model, $\mathbf{u}$ is the vector of parameters of the zipcode, ϵ is some random noise from the single house and $\mathbf{X}$ is the data.

Second level:

$$\epsilon(i) \sim N(0, \tau_\epsilon)$$
$$u(i) \sim N(0, \tau_u)$$
$$\beta(i) \sim N(0, \tau_\beta)$$

Third level (priors):

$$\tau_\epsilon \sim \sigma_\epsilon^{-2}$$
$$\sigma_\epsilon^2 \sim unif(0, 100)$$
$$\tau_\beta \sim \sigma_\beta^{-2}$$
$$\sigma_\beta^2 \sim inv.gamma(350, 5)$$
$$\tau_u \sim \sigma_u^{-2}$$
$$\sigma_u^2 \sim inv.gamma(350, 5)$$
$$\tau \sim \sigma^{-2}$$
$$\sigma^2 \sim inv.gamma(350, 5)$$

The implementation of the model with Jags is:

```
#####################################################################
#####################################################################
##JAGS parameters
y<- train$price
n <- nrow(train)                                    #size
m <- length(levels(as.factor(train$zipcode)))   #zones

#####################################################################
#####################################################################

model_code <- function(){
  # Likelihood:
  for (i in 1:N){
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- b0 + b1*waterfront[i] +  u[zipcode[i]] +
      b2 * sqft_living[i]  + eps[i]
  }

  # Second Level:
  b0 ~ dnorm(0, tau.b0) # b0
  b1 ~ dnorm(0, tau.b1) # b1
  b2 ~ dnorm(0, tau.b2) # b2
  for (i in 1:N) {
    eps[i] ~ dnorm(0, tau.eps)
  }
  for (j in 1:m){
    u[j] ~ dnorm(0, tau.u)
  }

  #Priors:
```

```
  tau.eps <- pow(sigma.eps, -2)
  sigma.eps ~ dunif(0,100)

  sigma.u <- 1/phi.u
  phi.u ~ dgamma(350, 5)
  tau.u <- 1 / (sigma.u * sigma.u) # convert to precision

  phi ~ dgamma(350, 5)
  sigma <- 1/phi
  tau <- 1 / (sigma * sigma) # convert to precision

  sigma.b0 <- 1/phi.b0
  phi.b0  ~ dgamma(350, 5)
  tau.b0  <- 1 / (sigma.b0 * sigma.b0) # convert to precision

  sigma.b1  <- 1/phi.b1
  phi.b1 ~ dgamma(350, 5)
  tau.b1 <- 1 / (sigma.b1 * sigma.b1) # convert to precision

  sigma.b2 <- 1/phi.b2
  phi.b2 ~ dgamma(350, 5)
  tau.b2 <- 1 / (sigma.b2 * sigma.b2) # convert to precision
}


model_data <- list(N = n, m = m, y = y, zipcode = train$zipcode,
                   waterfront = train$waterfront, sqft_living=train$sqft_living)

model_params <- c("b0", "b1", "b2", "mu", "tau", "u", "tau.b0", "tau.b1", "tau.b2",
                  "tau_a")

### Run the model
model_run  <- jags(
  data = model_data,
  parameters.to.save = model_params,
  model.file = model_code,
  n.chains = 3,
  n.iter = 10000,
  n.burnin = 2000,
  n.thin = 2
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3560
##    Unobserved stochastic nodes: 3584
##    Total graph size: 21815
##
## Initializing model
```

where we used 3 chains.

Results

Here the values for the βs.

```
head(round(model_run$BUGSoutput$summary, 2),10)
```

```
##                mean     sd      2.5%       25%       50%       75%     97.5%
## b0             3.88   0.03      3.82      3.86      3.88      3.90      3.93
## b1             0.25   0.02      0.22      0.24      0.25      0.26      0.28
## b2             0.67   0.01      0.65      0.66      0.67      0.68      0.69
## deviance -20133.42 379.24 -20880.03 -20383.73 -20133.17 -19880.48 -19377.58
## mu[1]          5.35   0.01      5.32      5.34      5.35      5.36      5.37
## mu[2]          5.78   0.01      5.75      5.77      5.78      5.79      5.81
## mu[3]          5.46   0.01      5.43      5.45      5.46      5.47      5.49
## mu[4]          5.36   0.01      5.34      5.35      5.36      5.37      5.39
## mu[5]          5.59   0.01      5.56      5.58      5.59      5.60      5.61
## mu[6]          5.97   0.01      5.94      5.96      5.97      5.98      6.00
##          Rhat n.eff
## b0          1 12000
## b1          1 12000
## b2          1 12000
## deviance    1 12000
## mu[1]       1 12000
## mu[2]       1 10000
## mu[3]       1  9800
## mu[4]       1 12000
## mu[5]       1 12000
## mu[6]       1 12000
```

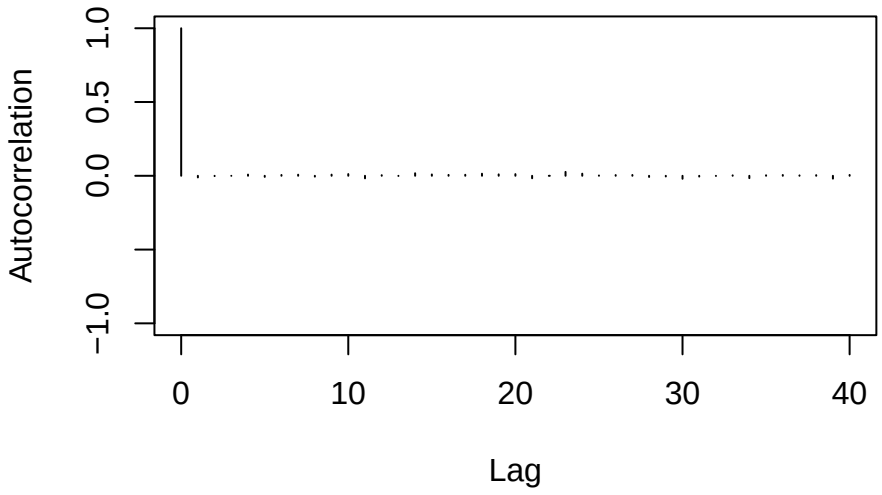
```
geweke.diag(as.mcmc(model_run$BUGSoutput$sims.array[,1,1:3]))
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      b0      b1      b2
## -0.7595  0.2958  0.3590
```

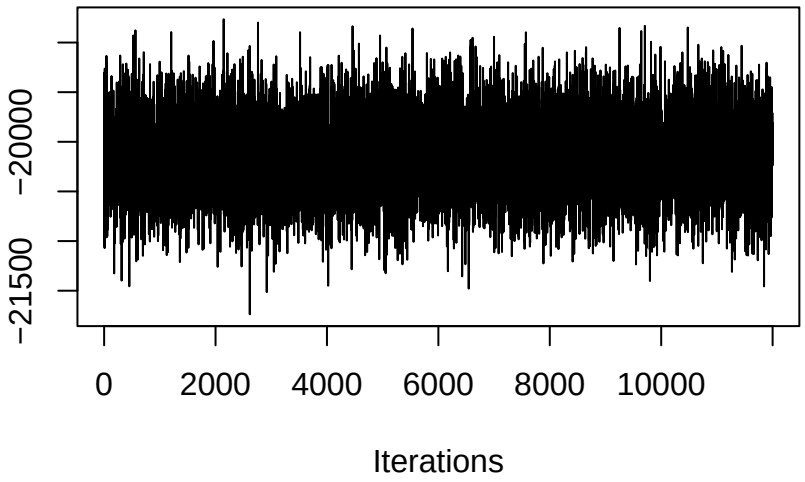
Diagnostics

Let's see the auto-correlation plot and some traceplots.

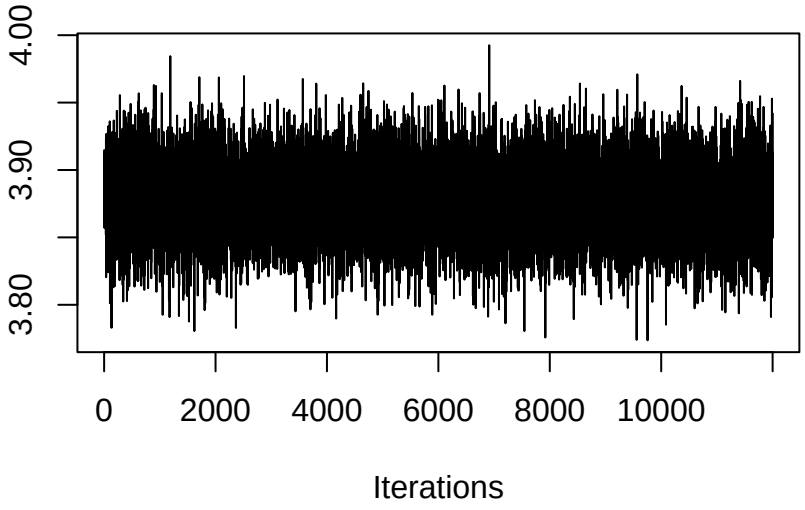
```
#par(mfrow=c(5,1))
coda::autocorr.plot(mcmc(model_run$BUGSoutput$sims.list$deviance))
```



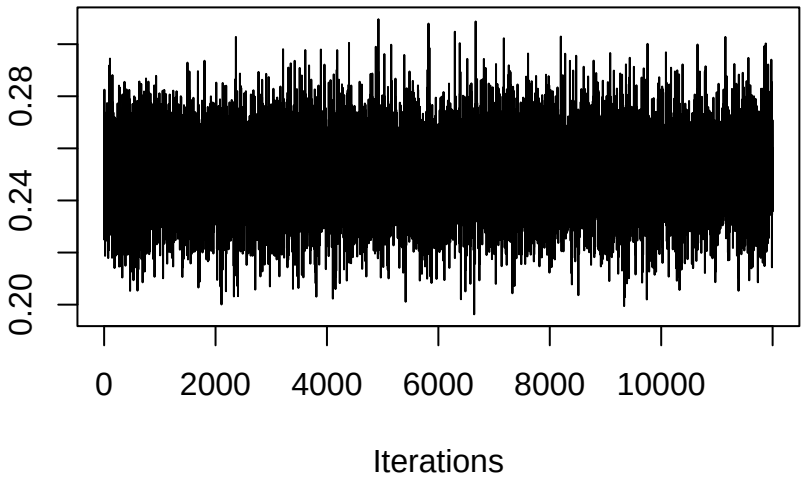

```
coda::traceplot(mcmc(model_run$BUGSoutput$sims.list$deviance))
```



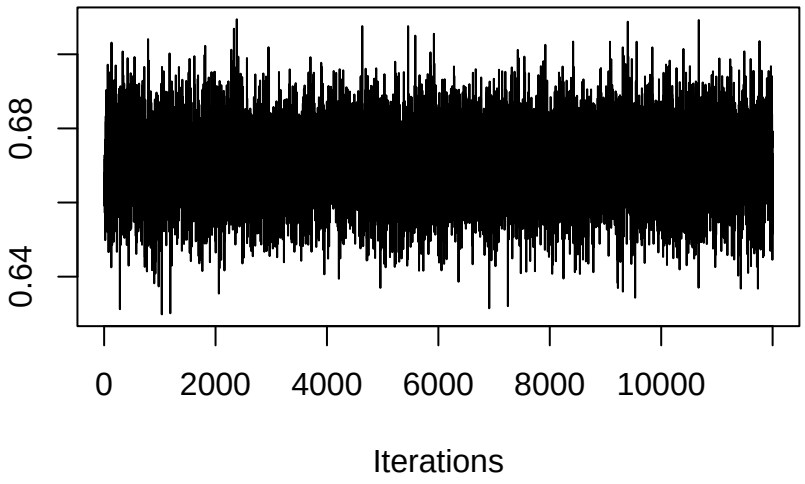

```
traceplot(mcmc(model_run$BUGSoutput$sims.list$b0))
```




```
traceplot(mcmc(model_run$BUGSoutput$sims.list$b1))
```




```
traceplot(mcmc(model_run$BUGSoutput$sims.list$b2))
```



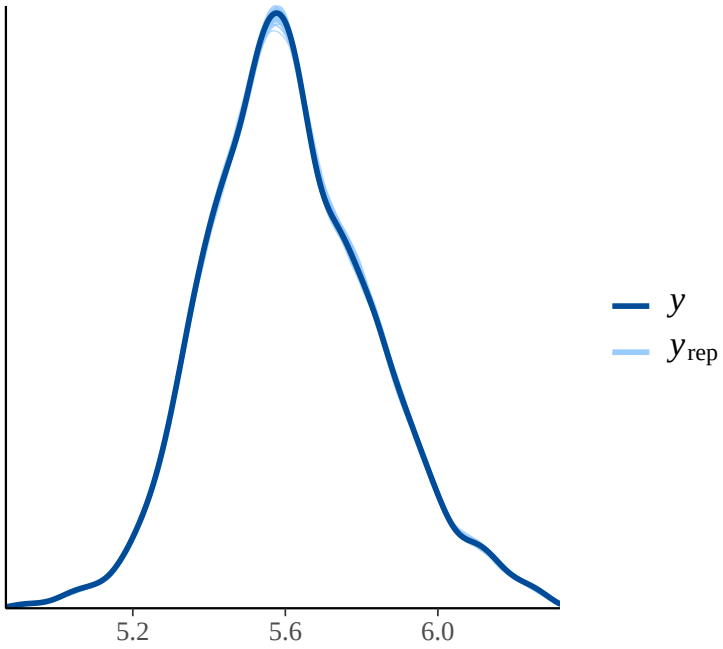
They seem good, in fact there's little auto-correlation and there's no evident pattern in the traceplots.

As for the model, first we extract the parameters and then we try to simulate some results using them:

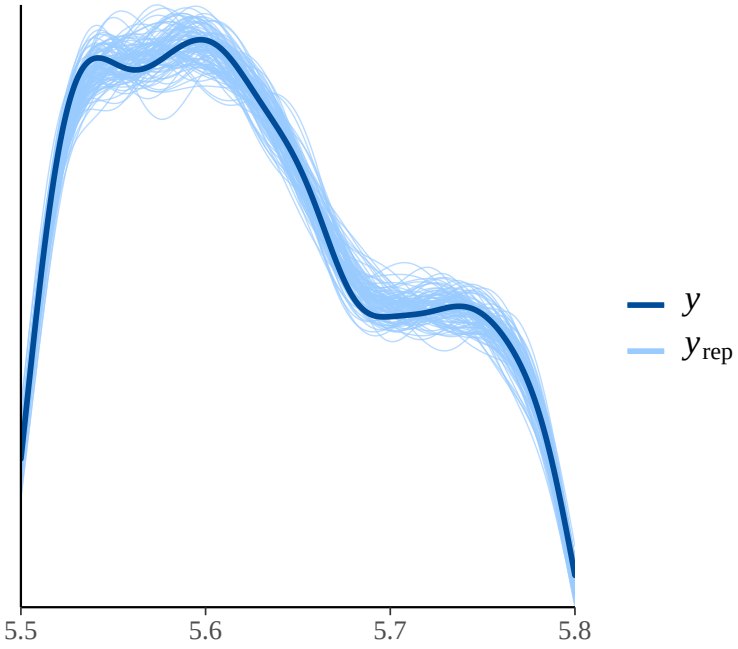
```
post.mu <- model_run$BUGSoutput$sims.list$mu
post.sigma2 <- (model_run$BUGSoutput$sims.list$tau)^-1
b0<-mean(model_run$BUGSoutput$sims.list$b0)
b1<-mean(model_run$BUGSoutput$sims.list$b1)
b2<-mean(model_run$BUGSoutput$sims.list$b2)
u<-colMeans(model_run$BUGSoutput$sims.list$u)


y2rep <- t(sapply(1:nrow(post.mu), function(x) rnorm(length(y), post.mu[x,],
                                                     post.sigma2[])))
```

In the next plots, we are checking how well our model fits the data. It performs very decently:

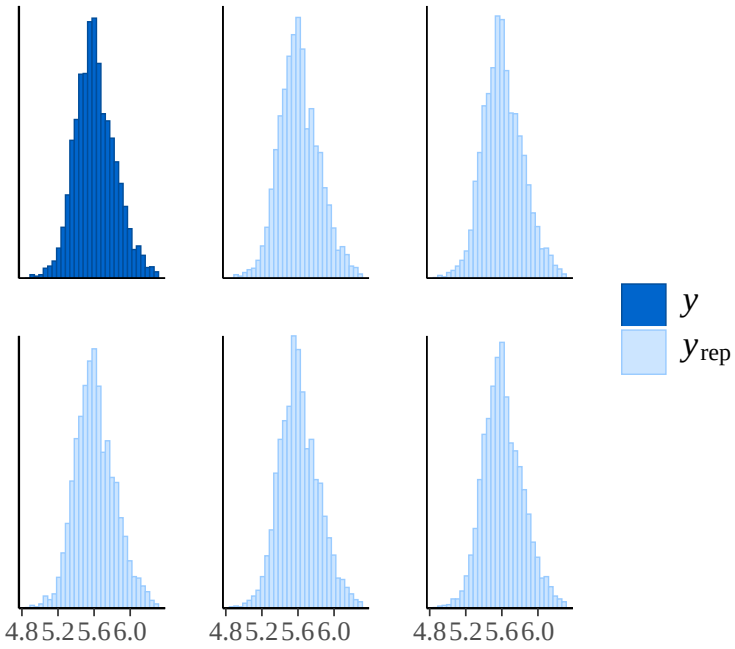
```
color_scheme_set("brightblue")

par(mfrow=c(3,2))
ppc_dens_overlay(y, y2rep[1:100, ])
```



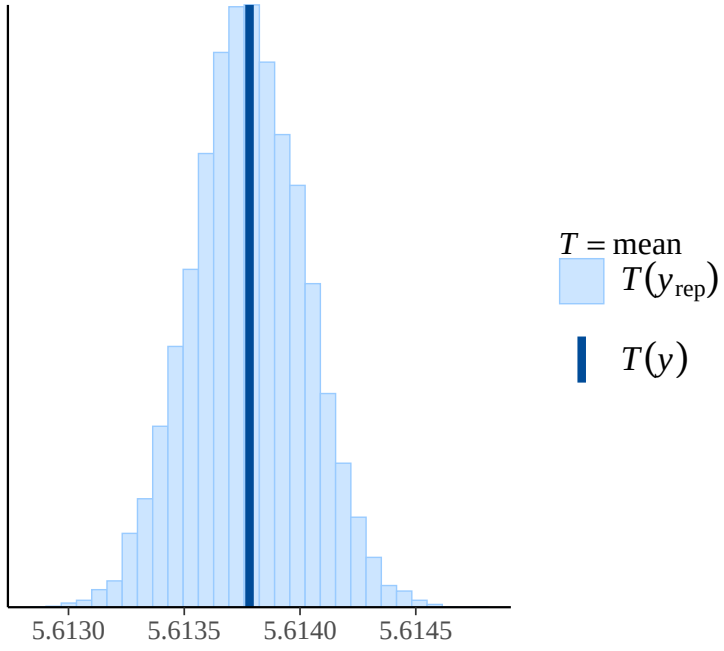

```
ppc_dens_overlay(y, y2rep[1:100, ]) + xlim(5.5,5.8) #zoom on the mean
```



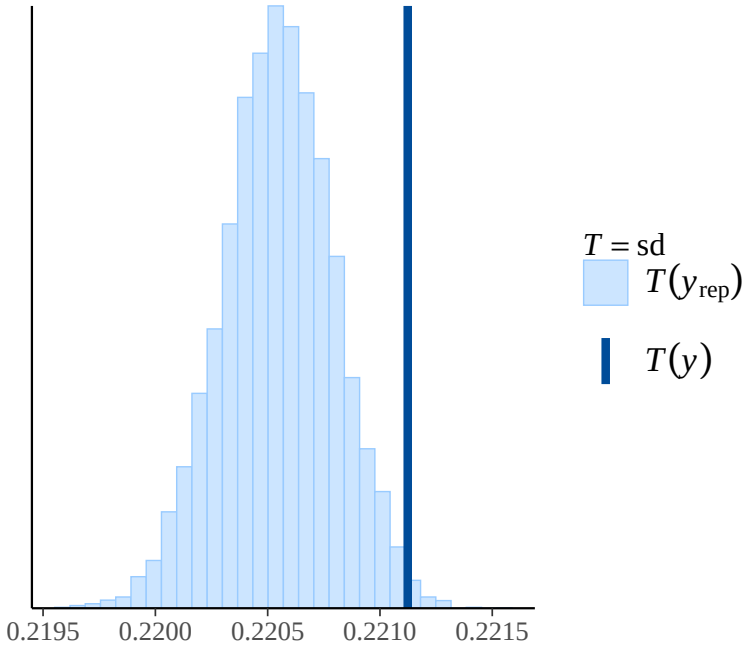

```
ppc_hist(y, y2rep[1:5, ])
```




```
ppc_stat(y, y2rep, stat = mean)
```

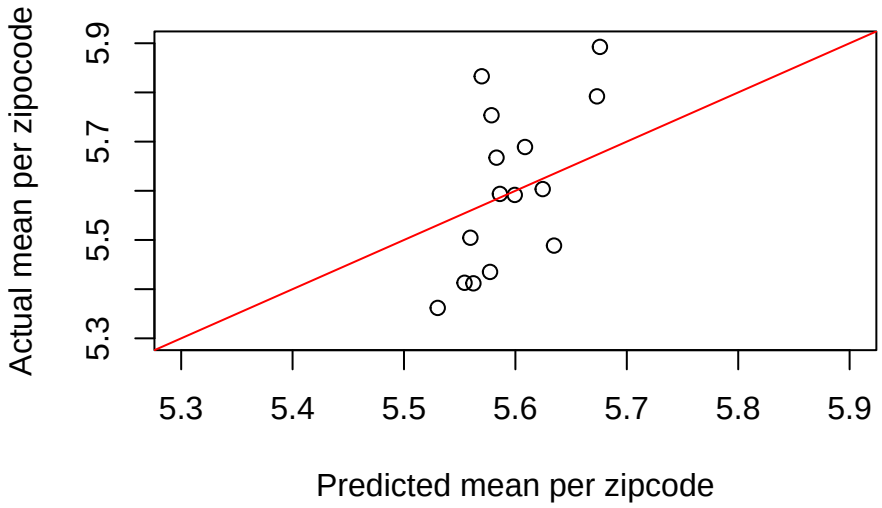



```
ppc_stat(y, y2rep, stat = sd)
```



We can see that there's a problem, though: the variance of the data is slightly underestimated! In fact, we can see it in the shrinkage effect on new data too:

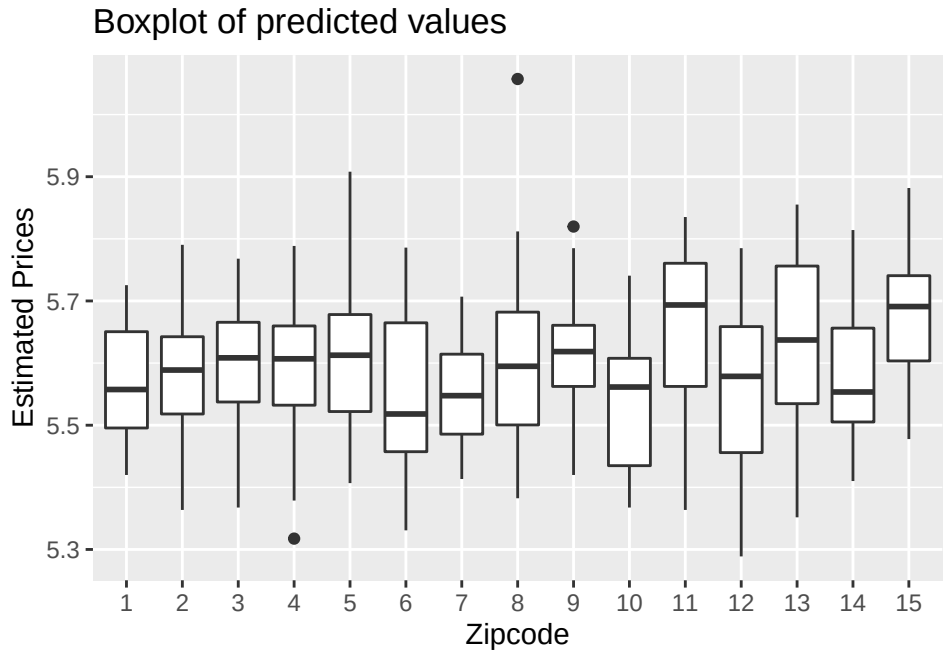
```
############################################################
#Shrinkage
############################################################
y.true <- test$price
y.pred <- NA

for (j in 1:nrow(test)){
  y.pred[j] <- b0 + b1*as.numeric(test$waterfront[j]) +b2*test$sqft_living[j]
  +u[test$zipcode[j]]
}

prova <-data.frame(prezzo=y.pred, zipcode=test$zipcode)
media_prev <-tapply(prova$prezzo, prova$zipcode, FUN=mean)
media_vera <-tapply(test$price, test$zipcode, FUN=mean)

plot(media_prev,media_vera,xlim=c(5.3,5.9),ylim=c(5.3,5.9),
     xlab = "Predicted mean per zipcode", ylab="Actual mean per zipocode")
abline(a=0,b=1, col="red")
```



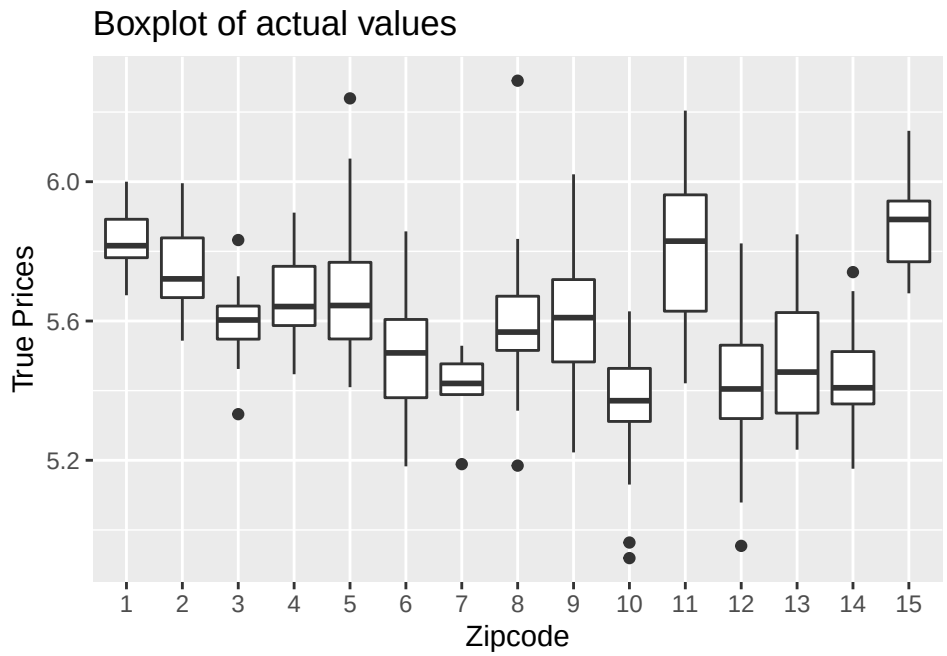
This plot shows the actual average prices per zipcode vs the predicted ones: for zones of the city with higher mean, our model tend to produce a lower prediction. On the other hand, it tends to overestimate the value of zipcodes with lower actual mean.

Here's a boxplot to confirm what we suspect: the variance between groups is poorly estimated.

```
par(mfrow=c(2,1))
ggplot(data=prova)+geom_boxplot(aes(x=as.factor(zipcode), y=prezzo ))+
  ggtitle("Boxplot of predicted values")+labs(x="Zipcode", y="Estimated Prices")
```



```
ggplot(data=test)+geom_boxplot(aes(x=as.factor(zipcode), y=price))+
  ggtitle("Boxplot of actual values")+labs(x="Zipcode", y="True Prices")
```

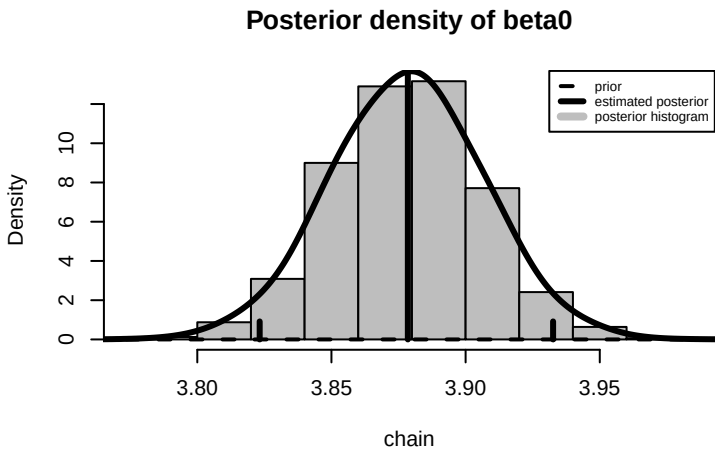


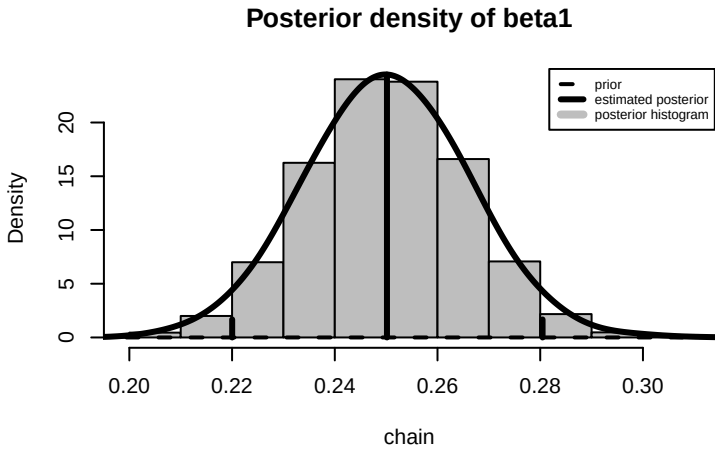
Finally, let's compute the Credible Intervals for some of the parameters of the model and let's visualize them with their posteriors.

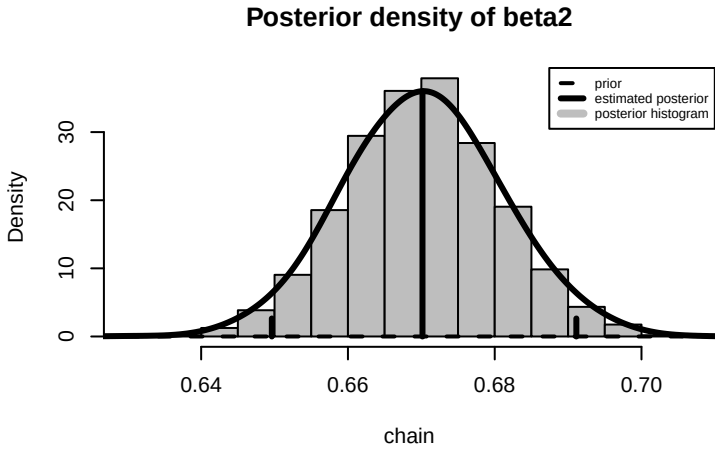
```
#############################################################################


beta_post <- model_run$BUGSoutput$sims.array[,1,1:3]

pre_beta<-dgamma(350, 5)

par(mfrow=c(3,1))
for(j in 1:3){
  chain <- beta_post[,j]
  prior_quant <- c(qnorm(0.025,0,sd=sqrt(1/pre_beta)),qnorm(0.975,0,sd=sqrt(pre_beta)))
  post_quant <- quantile(chain,prob=c(0.025,0.5,0.975))
  cat("The posterior median of beta",j-1," is ",post_quant[2],
      "\nThe 95% posterior credible interval for beta[",j-1,"] is (",
      post_quant[1],",",post_quant[3],")\n",sep="")
  post_dens <- approxfun(density(chain,adj=2))
  #Histogram
  hist(chain,freq=F,col="gray",xlim=range(chain),main=paste("Posterior density of beta",j-1,sep=""))
  ## overlap smooth posterior
  lines(density(chain,adj=2),col="black",lwd=3,lty=1)
  ## overlap prior
  curve(dnorm(x,0,sd=sqrt(1/pre_beta)),from=range(chain)[1],to=range(chain)[2],add=T,lty=2,lwd=2)
  ## add vertical segments in correspondence of the posterior credible bounds 95%
  segments(post_quant[1],0,post_quant[1],
           post_dens(post_quant[1]),col="black",lty=2,lwd=3)
```

```
  segments(post_quant[3],0,post_quant[3],
           post_dens(post_quant[3]),col="black",lty=2,lwd=3)
  segments(post_quant[2],0,post_quant[2],
           post_dens(post_quant[2]),col="black",lty=1,lwd=3)
  ## Add a legend
  legend("topright",c("prior","estimated posterior","posterior histogram"),
         lty=c(2,1,1),col=c("black","black","gray"),lwd=c(2,3,4),cex=0.6)

}
```

```
## The posterior median of beta0 is 3.878476
## The 95% posterior credible interval for beta[0] is (3.823251,3.932607)

## The posterior median of beta1 is 0.2501601
## The 95% posterior credible interval for beta[1] is (0.2200211,0.2804562)

## The posterior median of beta2 is 0.6701555
## The 95% posterior credible interval for beta[2] is (0.6496288,0.6911229)
```


Posterior density of beta0
prior
estimated posterior
posterior histogram
Density
0
2
4
6
8
10
3.80
3.85
3.90
3.95
chain


Posterior density of beta1
prior
estimated posterior
posterior histogram
Density
0
5
10
15
20
0.20
0.22
0.24
0.26
0.28
0.30
chain


Posterior density of beta2
prior
estimated posterior
posterior histogram
Density
0
10
20
30
0.64
0.66
0.68
0.70
chain