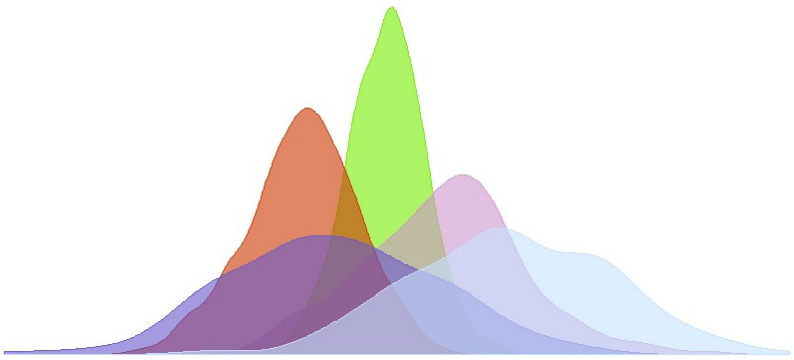
uc3m | Universidad Carlos III de Madrid

Bayesian report



Prof: *Micheal Wipe*

9th December 2019

Francesco Carbonera

100390180

Summary

Bayesian Generalized Linear Models

1. Description of the data

The dataset **birthwt**, which is contained in MASS package, is a set of 189 observations. The file has observations about factor risk associated with low child birth weight. Each observation is composed by 10 variables, but for the analysis we only need 9 of them. The variables refer to some factors visible on pregnant women.In particular, the variables are:

- **age** containing the mother's age at the childbirth;
- **lwt** containing mother's weight in pounds at last menstrual period;
- **race** containing a factorial variable with levels 1,2,3 that represent the mother's race (1 stands for white people, 2 for black and 3 for other);
- **ptl** containing a factorial representing the number of previous premature labours;
- **ht** containing a factorial representing the history of hypertension: 0 stands for no, 1 for yes;
- **ui** containing a factorial representing the presence of uterine irritability: 0 stands for no, 1 for yes;
- **ftv** containing a factorial representing the number of physician visits during the first trimester of pregnancy;
- **low** is the dependent variable and the values are 0 or 1. It represents the weight of the child at the birth: if it is 0 the weight is more than 2.5kg or else it is less.

The aim is to use this dataset to build a prediction model that will accurately classify if it is possible to know child conditions before the birth, knowing the mother's condition. Moreover, that could reveal which variables are more important to prevent low infant birth weight. For this purpose, I will use a GLM model (Binomial model) in a frequentist analysis and then in Bayesian one.

2. Preprocessing and Exploratory Data Analysis

Let's read the dataset from **MASS**. Firstly, I delete variable that will not used during the analysis, like **btw**. This variable represents body weight of the child at birth. I will use only the indicator function (**low**) of this variable.

```
data("birthwt")
b<-birthwt
b<-b[,-10]

dim(b)
```

```
## [1] 189   9
```

```
str(b)
```

```
## 'data.frame':    189 obs. of  9 variables:
##  $ low  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
##  $ race : int  2 3 1 1 1 3 1 3 1 1 ...
##  $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
##  $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
```

Secondly, as you can see using **str()** function, the variable **race**, **smoke**, **ptl**, **ht**,**ui** and **ftv** are factor variables, while in this datset they result as integer.Immediately a transformation is needed to make them factors. While, **age** and **lwt** will be used as continuous variables, even if their range is limited.

```
b$low<- as.factor(b$low)
b$race<- as.factor(b$race)
b$smoke<- as.factor(b$smoke)
b$ptl<- as.factor(b$ptl)
b$ht<- as.factor(b$ht)
b$ui<- as.factor(b$ui)
b$ftv<- as.factor(b$ftv)
```

As I told earlier, the aim of the project is to show different methodologies between frequentist and Bayesian analysis. To compare also the model results, I randomly select observations to create two distinct dataset (more less randomly: RN are always **pseudo** random number!). Thanks to that, we have a training set and a test set: one for analysis and other for predicting. The dataset contains 189 observations, I select for training set 180 observation, while the other 9 will be used for testing.

```
set.seed(1)
d<- sample(nrow(b),180, replace = F)
b_train <-b[d,]
b_test <- b[-d,]
```

The dataset used is **b_train**. It is made of 78% of discrete variables and 22% continuous one. Luckily, there are not missing observations.

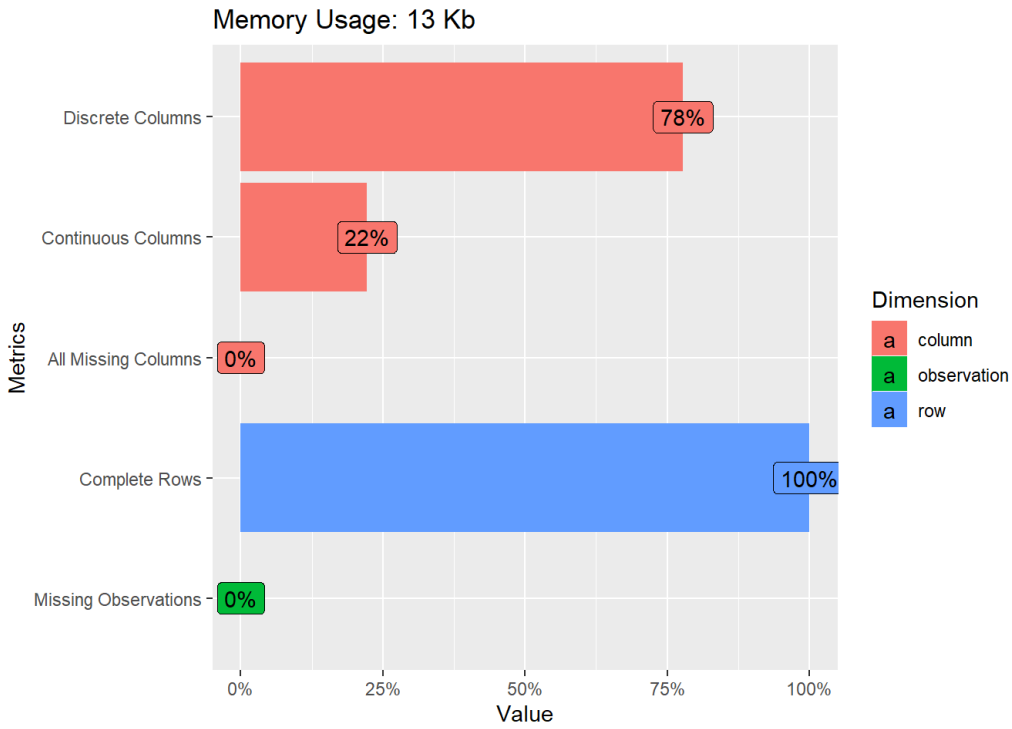
```
summary(b_train)
```

```
##  low          age             lwt        race   smoke   ptl     ht      ui
##  0:123   Min.   :14.00   Min.   : 80.0   1:93   0:110   0:152   0:168   0:152
##  1: 57   1st Qu.:19.00   1st Qu.:110.0   2:25   1: 70   1: 23   1: 12   1: 28
##          Median :22.00   Median :121.0   3:62           2:  4
##          Mean   :23.23   Mean   :130.4                  3:  1
##          3rd Qu.:26.00   3rd Qu.:141.2
##          Max.   :45.00   Max.   :250.0
##  ftv
##  0:96
##  1:43
##  2:29
##  3: 7
##  4: 4
##  6: 1
```

```
str(b_train)
```

```
## 'data.frame':    180 obs. of  9 variables:
##  $ low  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 1 1 ...
##  $ age  : int  28 21 24 25 23 30 22 23 32 25 ...
##  $ lwt  : int  250 100 116 105 130 107 120 123 121 241 ...
##  $ race : Factor w/ 3 levels "1","2","3": 3 3 1 3 2 3 1 3 3 2 ...
##  $ smoke: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
##  $ ptl  : Factor w/ 4 levels "0","1","2","3": 1 2 1 2 1 2 1 1 1 1 ...
##  $ ht   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 2 ...
##  $ ui   : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
##  $ ftv  : Factor w/ 6 levels "0","1","2","3",..: 6 5 2 2 2 3 2 1 3 1 ...
```

```
plot_intro(b_train)
```



The **age** range is 31, minimum age is 14 and maximum one is 45. For **lwt**, the range is 170 and minimum value is 80 while maximum one is 250.

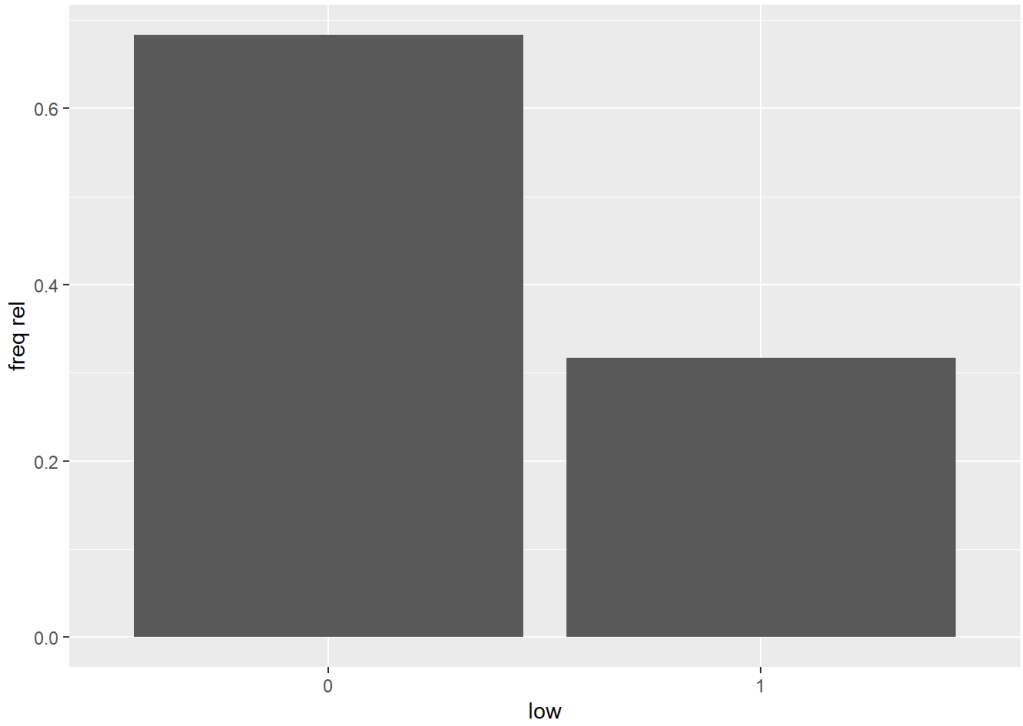
2.1 EDA on low variable

As showing by the table, **low** is clearly unbalance. There are more observations with 0 values than with 1.

```
b<-group_by(b_train,low)
summarize(b, count=n(),freq=n()/nrow(b_train))
```

```
## # A tibble: 2 x 3
##   low   count  freq
##   <fct> <int> <dbl>
## 1 0       123 0.683
## 2 1        57 0.317
```

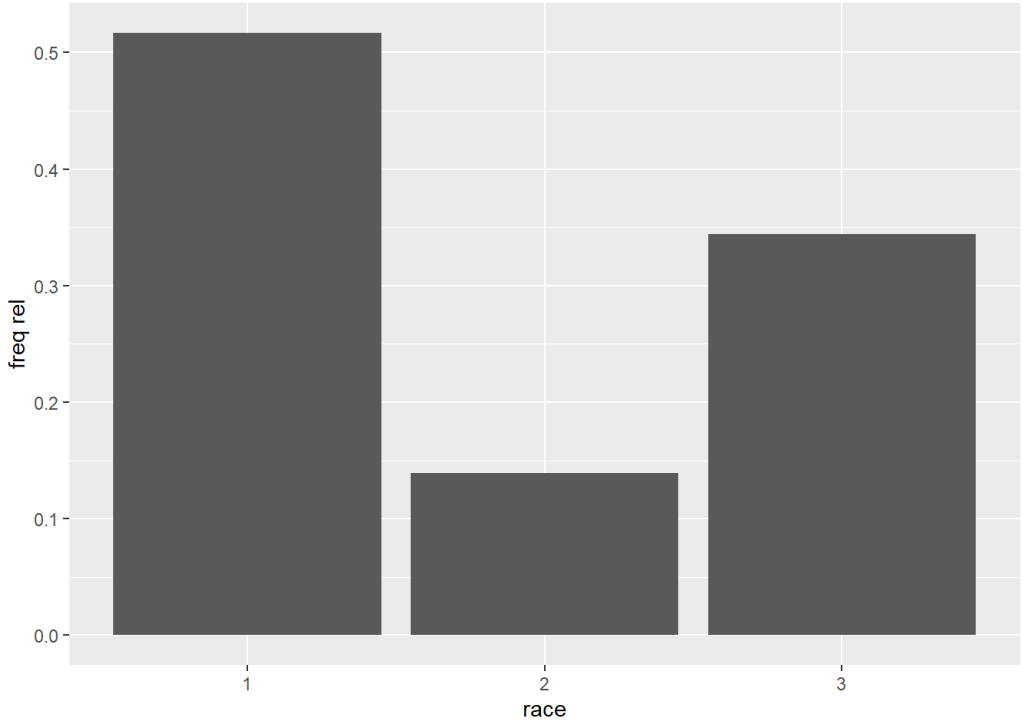
```
ggplot(data=b_train, aes(x=b_train$low)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))+ xlab("low")+ylab("freq rel")
```



2.2 EDA on other variables

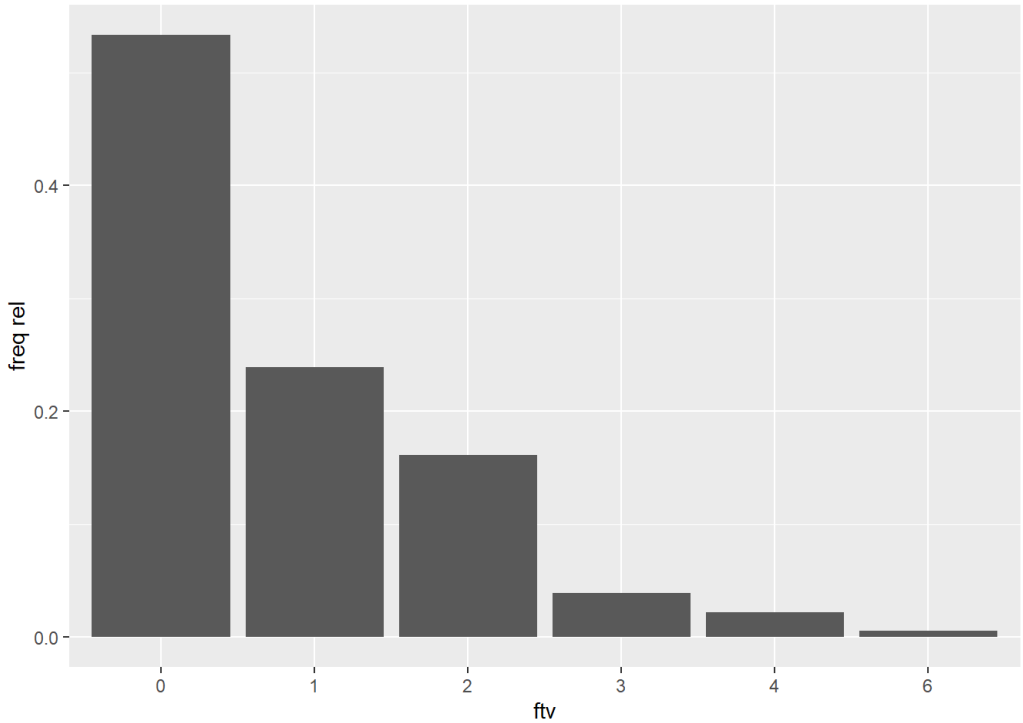
Now I will make EDA for each variable. For factorial variables I will show a barplot, while for others a histogram.

```
ggplot(data=b_train, aes(x=b_train$race)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))+ xlab("race")+ylab("freq rel")
```



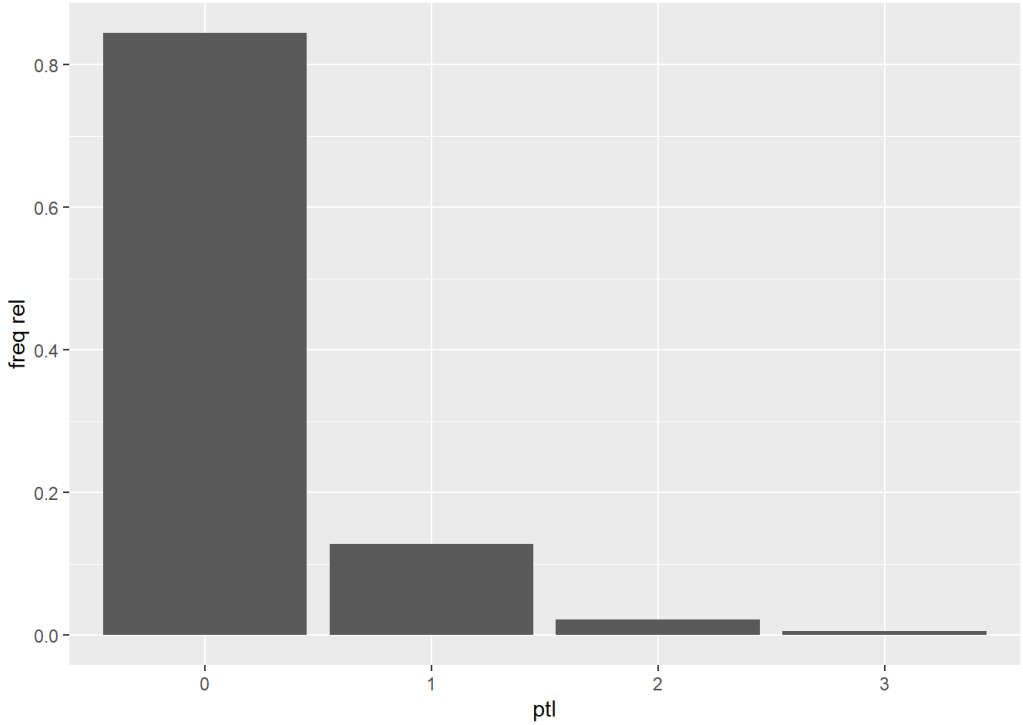
The variable **race** has 3 levels. The most frequent level represents white mothers while the black one contains less observations.

```
  ggplot(data=b_train, aes(x=b_train$ftv)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))+ xlab("ftv")+ylab("freq rel")
```



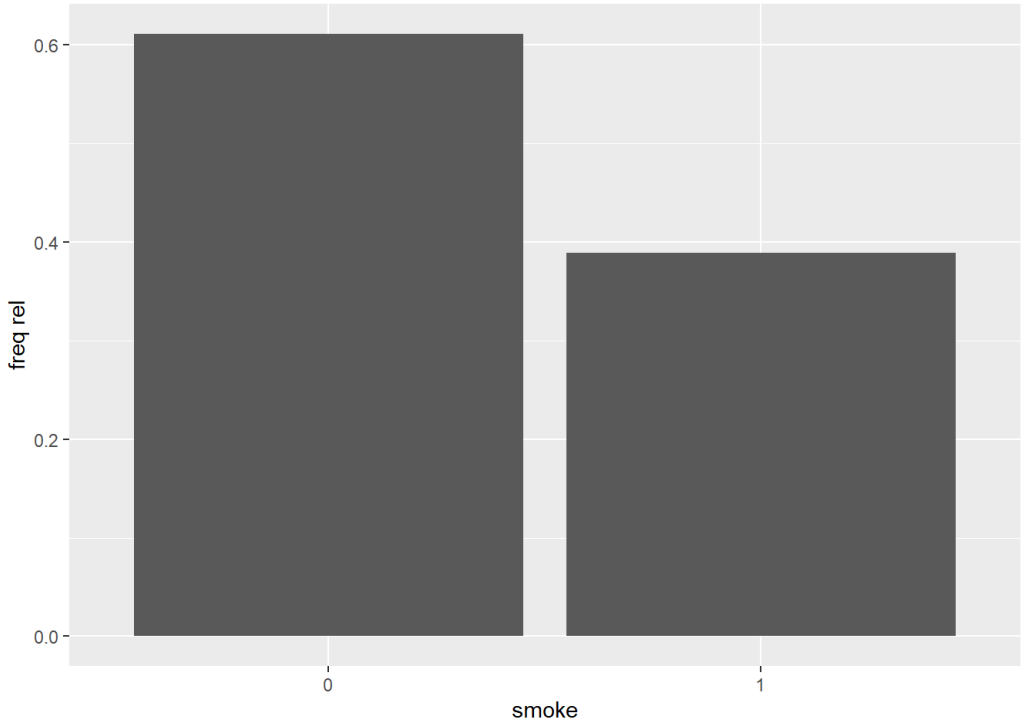
The level 0 contains more than 50% of observations. Observations decrease for high levels: level 6, representing 6 physician visits, only has 1 observation.

```
ggplot(data=b_train, aes(x=b_train$ptl)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))+xlab("ptl") +ylab("freq rel")
```



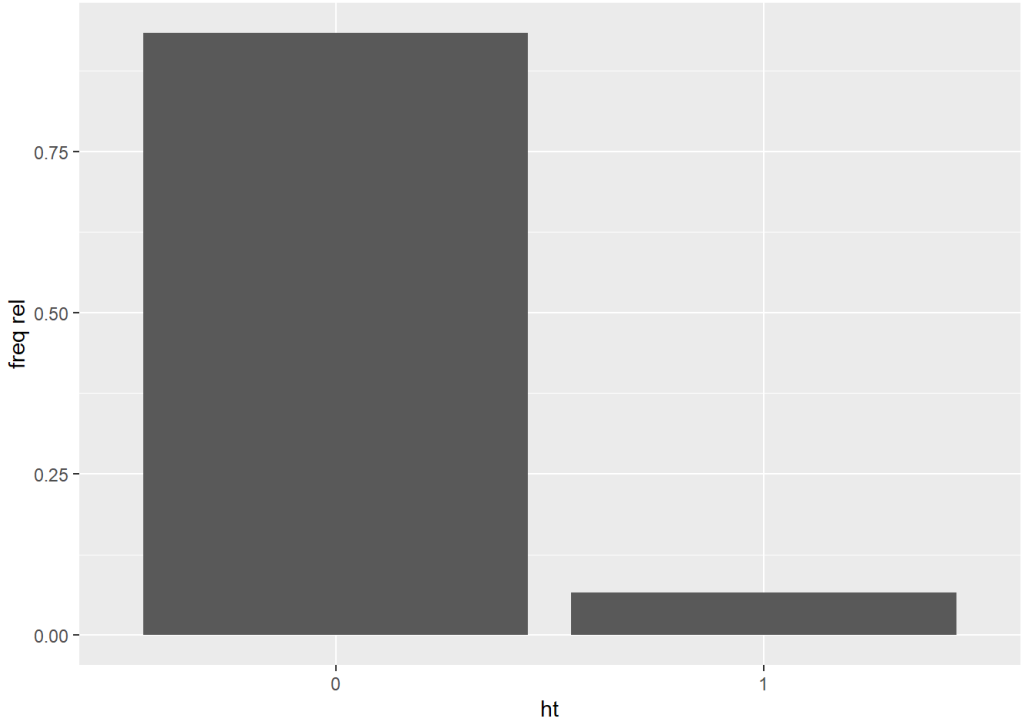
This plot shows us the distribution of **ptl**. However, mothers with more than 0 premature labours are less than 0.2 (level 3 only has 1 observation).

```
ggplot(data=b_train, aes(x=b_train$smoke)) +
geom_bar(aes(y = (..count..)/sum(..count..)))+ xlab("smoke")+ylab("freq rel")
```



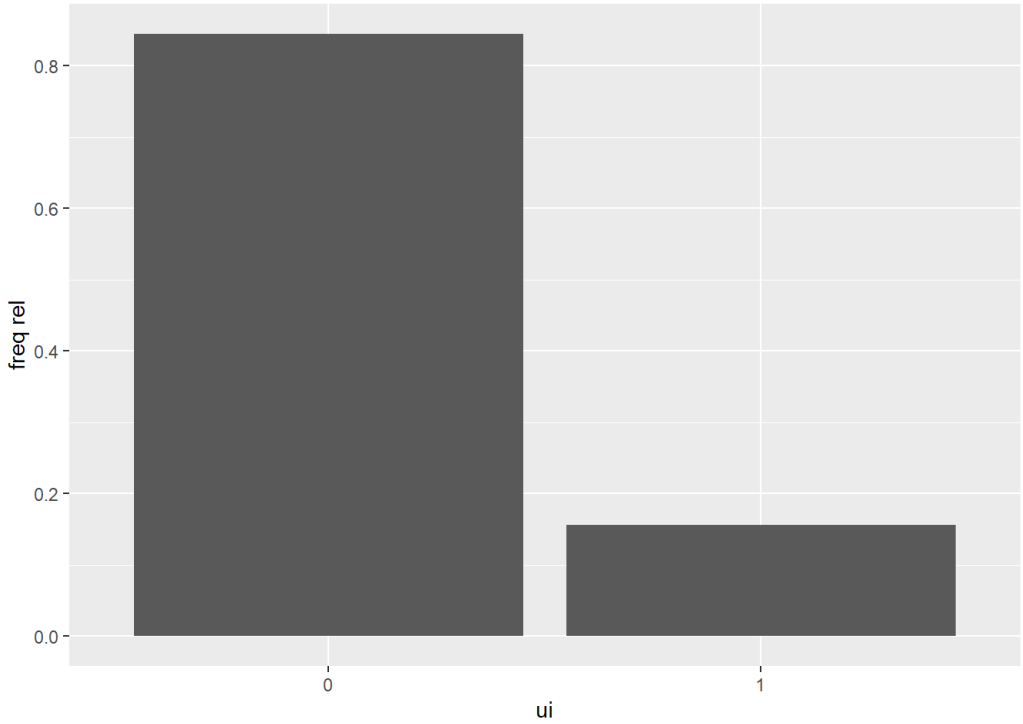
The **smoke** barblot reveals that mothers who smoke are less than those who do not: they are less than 0.4. By the way, this variable is the most balanced one.

```
ggplot(data=b_train, aes(x=b_train$ht)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))+xlab("ht")+ ylab("freq rel")
```



The plot shows the variable **ht**. It is unbalanced: there are more mothers without hypertension (more less 90%).

```
ggplot(data=b_train, aes(x=b_train$ui)) +
geom_bar(aes(y = (..count..)/sum(..count..)))+ xlab("ui")+ylab("freq rel")
```

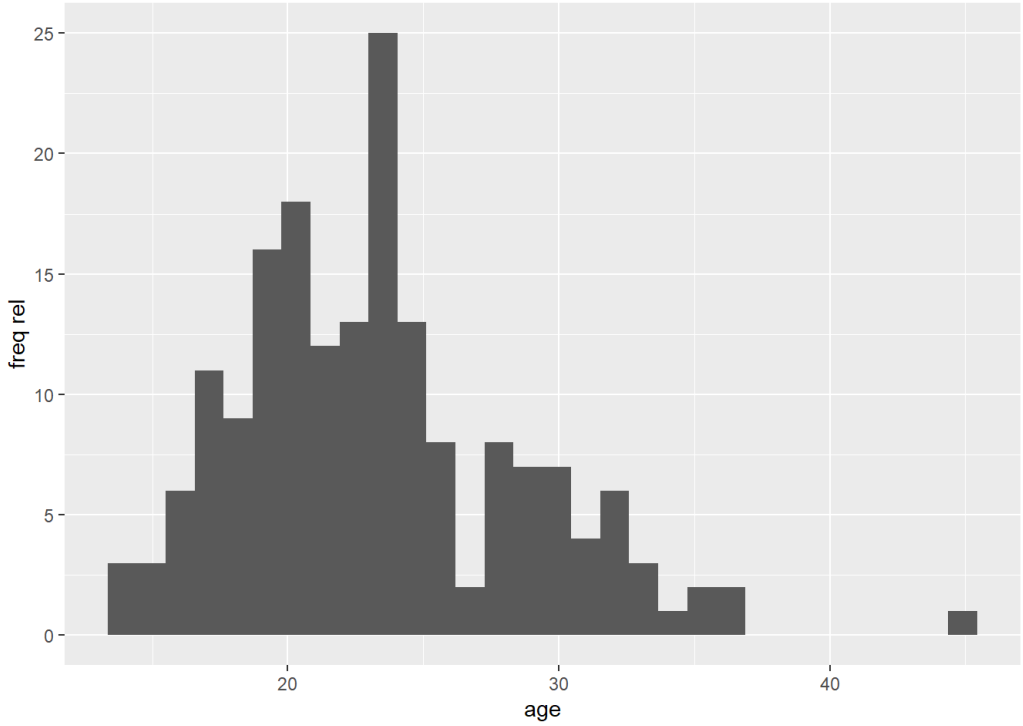


The absence of uterine irritability is the major class: more than 80%.

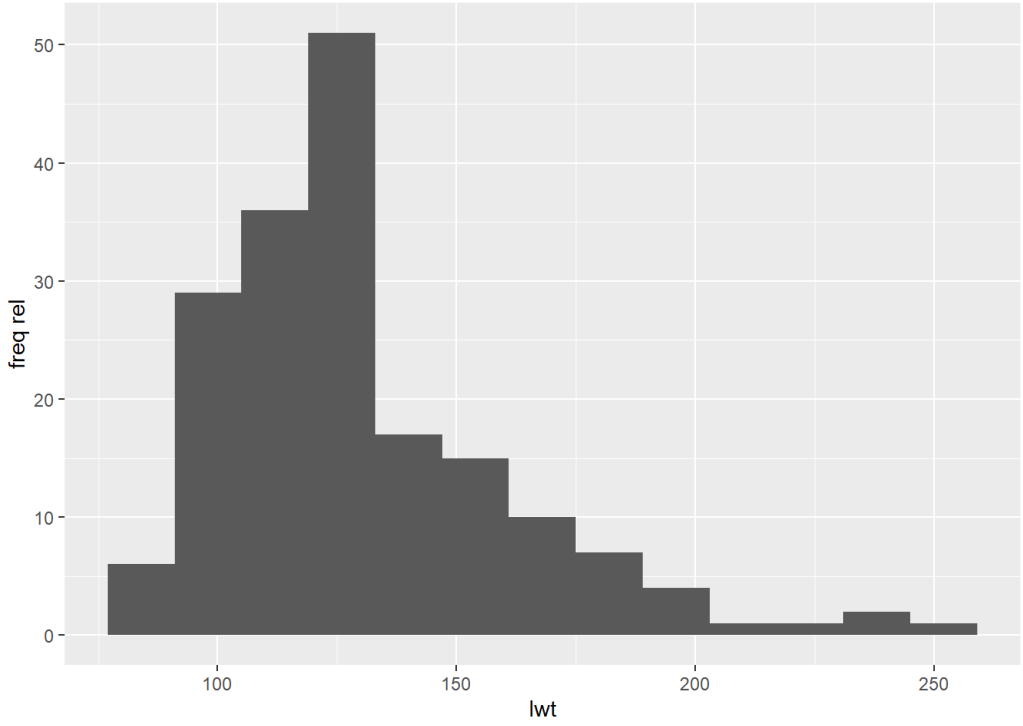
For continuous variables, the histogram reveals symmetric distribution for **age**, while **lwt** is slightly asymmetrical on the left.

```
par(mfrow=c(2,1))
  ggplot(data=b_train, aes(x=b_train$age)) +
  geom_histogram()+ xlab("age")+ylab("freq rel")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



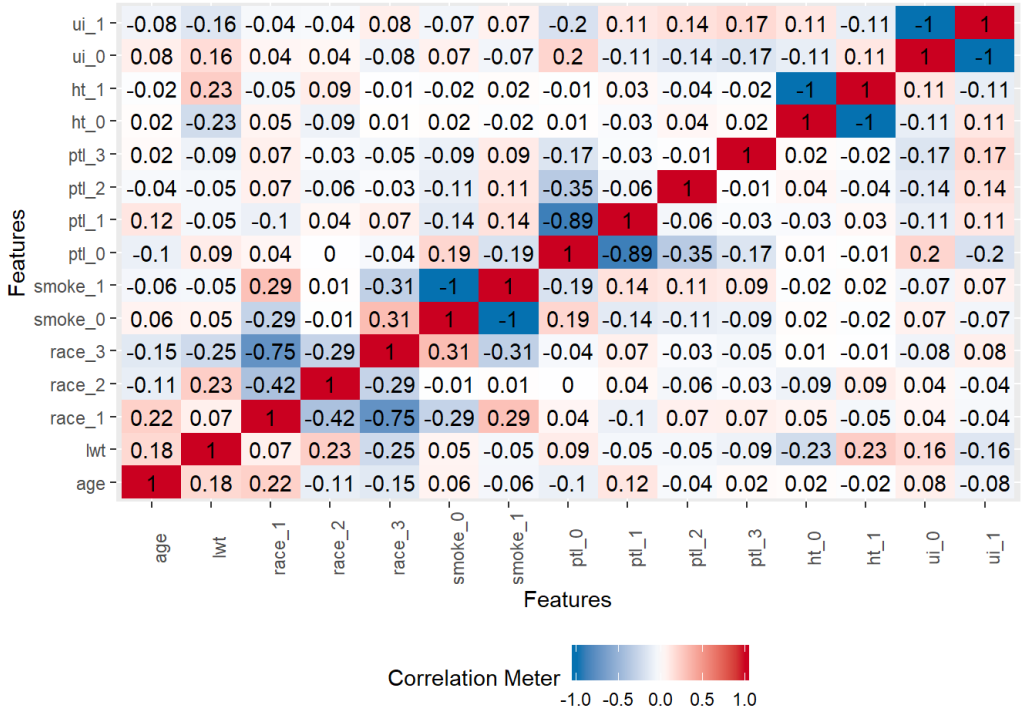

```
ggplot(data=b_train, aes(x=b_train$lwt)) +
geom_histogram(binwidth = 14 )+ xlab("lwt")+ylab("freq rel")
```



The following plot shows the correlation between covariates. We are interested to see if there is correlation between variables because it could simplify analysis and regression part. This does not reveals much information. However, correlation is high between different levels of the same variable: this is so obvious (if you do not smoke you will have corr= -1 with smoking variable)!

```
b_train2<-b_train[ ,-1]
plot_correlation(b_train2,maxcat = 5L)
```

```
## 1 features with more than 5 categories ignored!
## ftv: 6 categories
```

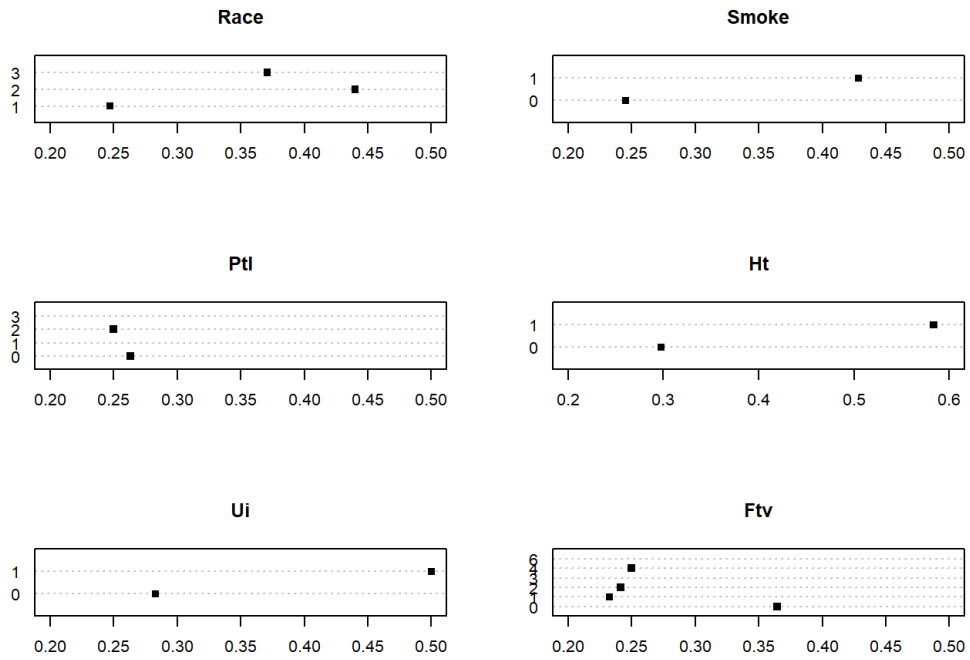


Furthermore, only correlations between variables with less than 6 levels are shown. Only **ftv** is not showed.

2.3 EAD between low and other variables

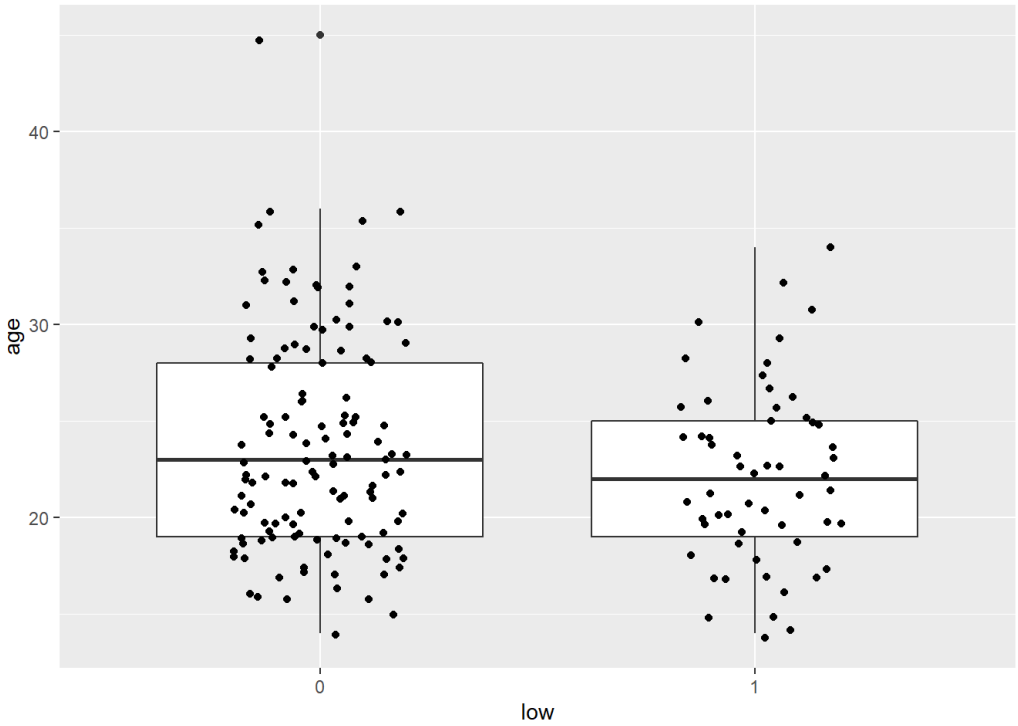
I analyze **low** distribution for each variable, using **dotchart**.

```
par(mfrow=c(3,2))
b_train$low <- as.numeric(b_train$low)-1
d<-group_by(b_train,race)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Race",xlim = c(0.2,0.5), labels = e$race)

d<-group_by(b_train,smoke)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Smoke",xlim = c(0.2,0.5), labels = e$smoke)

d<-group_by(b_train,ptl)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Ptl",xlim = c(0.2,0.5), labels = e$ptl)

d<-group_by(b_train,ht)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Ht",xlim = c(0.2,0.6), labels = e$ht)

d<-group_by(b_train,ui)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Ui",xlim = c(0.2,0.5), labels = e$ui)

d<-group_by(b_train,ftv)
e<-summarize(d, count=n(),sum=sum(low), freq=sum/count)
dotchart(e$freq, pch=15,, main="Ftv",xlim = c(0.2,0.5), labels = e$ftv)
```

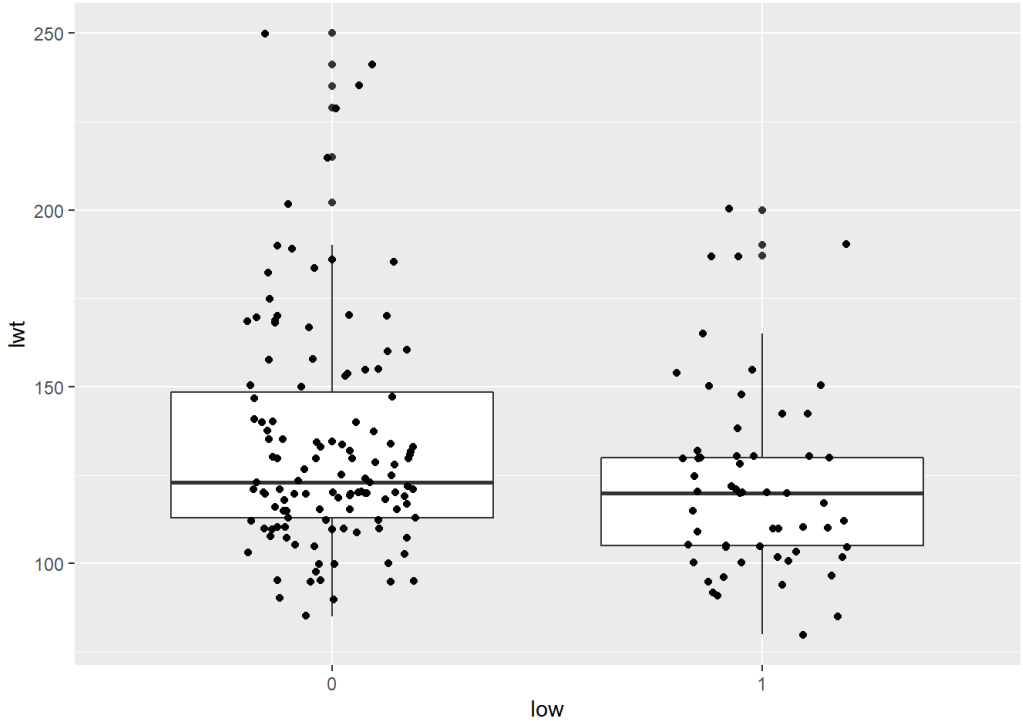


As you can see,the difference between **ptl** and **ftv** levels is very small, that should suggest these variable could not have significance in regression.

```
ggplot(b_train, aes(x=low, y=age)) + geom_boxplot()+ geom_jitter(shape=16, position=position_jitter(0.2))
```




```
ggplot(b_train, aes(x=low, y=lwt)) + geom_boxplot()+ geom_jitter(shape=16, position=position_jitter(0.2))
```



The boxplot of **age** suggests the values between two levels of **low** have similar distribution, while for **lwt** there seems to be significant difference between two levels. A t-test between mean difference could show something interesting. At level 5%, the difference between **age** mean of two levels is not significative, while it is for **lwt**.

```
t.test(age~low, b_train)
```

```
##
##  Welch Two Sample t-test
##
## data:  age by low
## t = 1.8927, df = 131.63, p-value = 0.06059
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.06719385  3.04451520
## sample estimates:
## mean in group 0 mean in group 1
##        23.69919        22.21053
```

```
t.test(lwt~low, b_train)
```

```
##
##  Welch Two Sample t-test
##
## data:  lwt by low
## t = 2.4135, df = 130.34, p-value = 0.01719
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   1.989973 20.081914
## sample estimates:
## mean in group 0 mean in group 1
##        133.8780        122.8421
```

3. GLM with frequentist methods

To analyze the dependent variable, the method we will use is a logistic regression. It is used to model probability of bernoullian variable (two possible values). To estimate this model we use an exponential family distribution $Y \sim EF$. The distribution is $f(x;\theta) = h(x)c(\theta)exp(w(\theta)t(x))$. A bernoulli variable Y assigns probability measure p to y=1 and 1-p to y=0. The density function is $p(y \mid p) = p^y(1-p)^{1-y} = exp\left\{log(\frac{p}{1-p})y + log(1-p)\right\}$. A GLM is characterized by:

- the variable needs to belong in exponential family $Y_i \sim EF$
- the covariate affect dependent variable in linear way. The function takes the name of linear predictor $\eta_i = x_i^T\beta = \sum \beta_i x_i$
- existence of a function, called link function, that connects mean to the linear predictor $\mu_i = g(\eta_i)$, so $\eta_i = g(\mu_i)^{-1}$.

For our problem, we use logistic regression and the Y distribution is a bernoulli one:

- the model I use is a Bernoulli one and belong EF $Y_i \sim Bern(p_i)$;
- I set a linear predictor $\eta_i = x_i^T\beta$;
- I set a link function to connect the linear predictor. In this case I use the canonic one: logit function. So, we have $logit(p_i) = log(\frac{p_i}{1-p_i}) = x_i^T\beta$. I choose the canonic link function neverthless there are other two different link functions:
  - probit funcion: $\Phi(p_i)^{-1} = x_i^T\beta$;
  - complementary log-log (cloglog): $log(-log(1-p_i)) = x_i^T\beta$.

3.1 Some simple models

- **Model with only intercept**

For the three models, I use few variables only to show principal effect of them. The first one has only the intercept. It is like $logit(p_i) = \beta_0$. If you do not select any link function, the canonic one is setted as default.

```
fit0=glm(low~1,family=binomial,data=b_train)
summary(fit0)
```

```
##
## Call:
## glm(formula = low ~ 1, family = binomial, data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8727  -0.8727  -0.8727   1.5165   1.5165
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.7691     0.1602    -4.8 1.59e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 224.76  on 179  degrees of freedom
## AIC: 226.76
##
## Number of Fisher Scoring iterations: 4
```

I obtain a coefficient of $\hat{\beta}_0 = -0.7691$. This coefficient is significant, but the model is too simple to explain relationship.

- Model with only **age**

```
fitA <- glm(low ~ age, family=binomial,
           data=b_train)
summary(fitA)
```

```
##
## Call:
## glm(formula = low ~ age, family = binomial, data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0665  -0.9126  -0.7731   1.3983   1.7969
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.5216     0.7507   0.695   0.4872
## age          -0.0563     0.0324  -1.737   0.0823 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 221.57  on 178  degrees of freedom
## AIC: 225.57
##
## Number of Fisher Scoring iterations: 4
```

You can see the variable **age** is not significant at level 5%. It was suggested also by t.test seen later in EDA part.

- Model with **age + lwt**

```
fitB <- glm(low ~ age+lwt, family=binomial,
            data=b_train)
summary(fitB)
```

```
##
## Call:
## glm(formula = low ~ age + lwt, family = binomial, data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1555  -0.9117  -0.7483   1.3349   2.0537
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.803026   1.009914   1.785   0.0742 .
## age         -0.045226   0.033114  -1.366   0.1720
## lwt         -0.012004   0.006203  -1.935   0.0530 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 217.37  on 177  degrees of freedom
## AIC: 223.37
##
## Number of Fisher Scoring iterations: 4
```

The variable **lwt** is not significant to explain **low** variable in a model with **age** effect.

- **Model selection**

I decide to use **step** function to choose the best model minimizing the AIC value (I could also choose the function **stepAIC()** from **MASS** library). The stepwise algorithm consists in using all variables and trying to generate a lot of combination of models. At the end, it will be show the best model chosen, according to AIC value. Namely, the model which minimizes AIC value. The AIC is a value which gives importance to the number of parameters used (k) and to the maximum value of the log likelihood function $log(\hat{L})$: $AIC = 2k - 2log(\hat{L})$.

```
fitSTEP <- step(fitB, scope= ~ age+lwt+race+smoke+ptl+ht+ui+ftv, direction="both")
```

```
summary(fitSTEP)
```

```
##
## Call:
## glm(formula = low ~ lwt + ptl + ht + smoke + ui + race, family = binomial,
##     data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9726  -0.7753  -0.5019   0.8876   2.1994
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.046720   1.011136   0.046  0.96315
## lwt          -0.017190   0.007254  -2.370  0.01780 *
## ptl1          1.570982   0.528627   2.972  0.00296 **
## ptl2         -0.595197   1.247350  -0.477  0.63324
## ptl3        -14.900856 882.743539  -0.017  0.98653
## ht1           1.843065   0.716573   2.572  0.01011 *
## smoke1        1.015314   0.422272   2.404  0.01620 *
## ui1           0.905761   0.481690   1.880  0.06006 .
## race2         1.238179   0.546172   2.267  0.02339 *
## race3         0.702818   0.465729   1.509  0.13128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 182.22  on 170  degrees of freedom
## AIC: 202.22
##
## Number of Fisher Scoring iterations: 13
```

This is the final model chosen by step algorithm:
$logit(p_i) = \beta_0 + \beta_{lwt}x_{lwti} + \beta_{ptl1}\delta_{ptl1i} + \beta_{ptl2}\delta_{ptl2i} + \beta_{ptl3}\delta_{ptl3i} + \beta_{ht1}\delta_{ht1i} + \beta_{smoke1}\delta_{smoke1i} + \beta_{ui1}\delta_{ui1i} + \beta_{race2}\delta_{race2} + \beta_{race3}\delta_{race3i}$
where δ_{ki} is a dummy variable with value 1 if variable k=1 else 0, while β_{ki} are for continuos variables. Moreover, the stepwise algorithm also selects variable like **ptl** whose coefficients are not significant. So to choose the best model in frequentist way, I decide to delete **ptl** and look the result.

```
fitfinal <- glm(low ~ lwt+ht+smoke+ui+race, family=binomial,
            data=b_train)
summary(fitfinal)
```

```
##
## Call:
## glm(formula = low ~ lwt + ht + smoke + ui + race, family = binomial,
##     data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6844  -0.8441  -0.5275   0.9681   2.1627
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.034051   0.954603  -0.036  0.97155
## lwt         -0.015965   0.006845  -2.332  0.01969 *
## ht1          1.810500   0.687649   2.633  0.00847 **
## smoke1       1.131452   0.400555   2.825  0.00473 **
## ui1          0.877135   0.451636   1.942  0.05212 .
## race2        1.343807   0.529490   2.538  0.01115 *
## race3        0.881632   0.441990   1.995  0.04608 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 194.31  on 173  degrees of freedom
## AIC: 208.31
##
## Number of Fisher Scoring iterations: 4
```

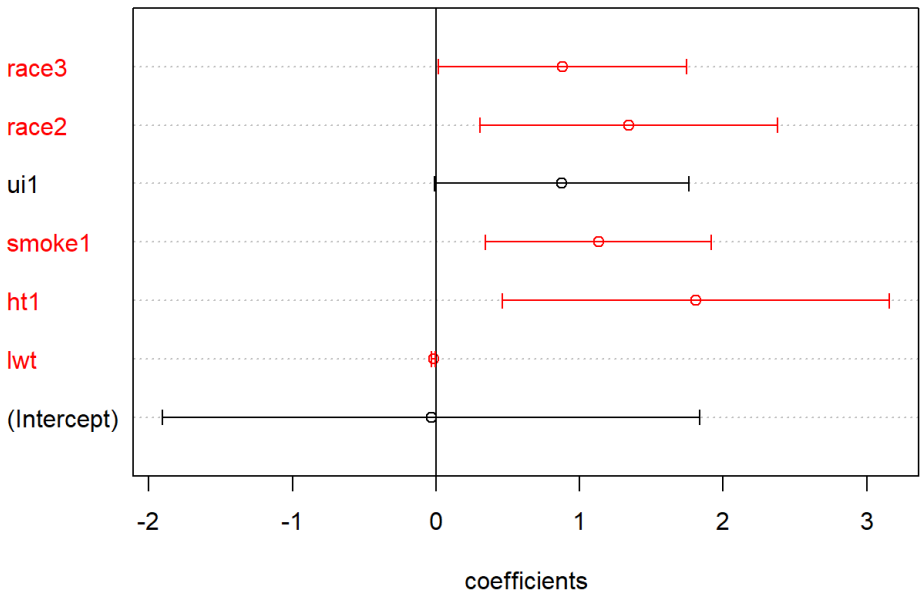
It is possible to explain variable effects on the linear predictor $\eta_i = x_i^T\beta$:

- Smoking increases the predictor by 1.13;
- Being black increases the predictor by 1.34, while other kind of race (not the white one) increases by 0.88;
- Suffering of hypertension increases the predictor by 1.81;
- Suffering of uterine irritability increases the predictor by 0.87;

However, the effects do not impact directly on the probability but on the link function. You keep in mind that the link function is $log(\frac{p_i}{1-p_i}) = x_i^T\beta$ and so to have p_i we have to take this transformation $p_i = \frac{e^{x_i^T\beta}}{1-e^{x_i^T\beta}}$

It is possible to create a dotchar to visualize 95% confidence interval for $\hat{\beta}_i$.

```
 par(mfrow=c(1,1))
coef <- summary(fitfinal)
up<-coef$coefficients[,1]+1.96*coef$coefficients[,2]
low <-coef$coefficients[,1]-1.96*coef$coefficients[,2]
signif=!(0<up & 0>low)
col=c("black","red")[1+signif]
dotchart(coef$coefficients[,1],xlim = range(low,up), xlab ="coefficients", col=col)
abline(v=0)
arrows(low,1:nrow(coef$coefficients),up, 1:nrow(coef$coefficients), length=0.05, angle=90, code=3,col=col)
```



3.2 Prediction

Using **carsred_test** dataset, it is possible to predict **low** :

```
pred <- predict(fitfinal,type = "response", newdata = b_test)
pred
```

```
##       116       118       125       163       189       208       214        24
## 0.3788891 0.4159478 0.2927151 0.3975494 0.2577212 0.2557465 0.2265592 0.2712340
##        32
## 0.3604790
```

4. GLM with Bayesian methods

The Bayesian method consists in giving a priori-distribution to parameters reflecting uncertainty about them. Then the MCMC (Markov chain Monte Carlo) techniques consist of simulating sampling to approximate posterior distribution of parameters by Monte Carlo with Markov chain. The simulation continues to generate random values and estimates the properties of a distribution by examining them (Monte Carlo) under some rules to determine good parameters. Then,the idea of Markov chain property of MCMC is that the random samples are generated by a special sequential process. Each random sample is used as a step system to generate the next random sample. The property of the chain is that, while each new sample depends on the one before it, new samples do not depend on any samples before the previous one. So, the difference with frequentist analysis is that we give a distribution representing beliefs on our parameters (prior distribution). Then, the parameters are estimating and chosen to maximize likelihood function. Thus, the posterior distribution is determined combining prior distribution and likelihood function.

Let us run the code, using default values:

- **burnin** represents the number of iterations are thrown away at the beginning of MCMC run. In finding convergence of parameters the first iterations could be very random and to limit the number of posterior samples they are deleted that also saving computer memory;
- **mcmc** represents the number of iterations for the sampler;
- **thin** is used to reduce autocorrelation in the resulting samples. Autocorrelation tends to decrease when the lag increases. Setting this parameter, a sample is produced by the MCMC vector considering only every thin value iterations of the chain;
- **beta.start** the starting value of β vector;
- **b0** the prior mean of β;
- **B0** the prior precision of β.

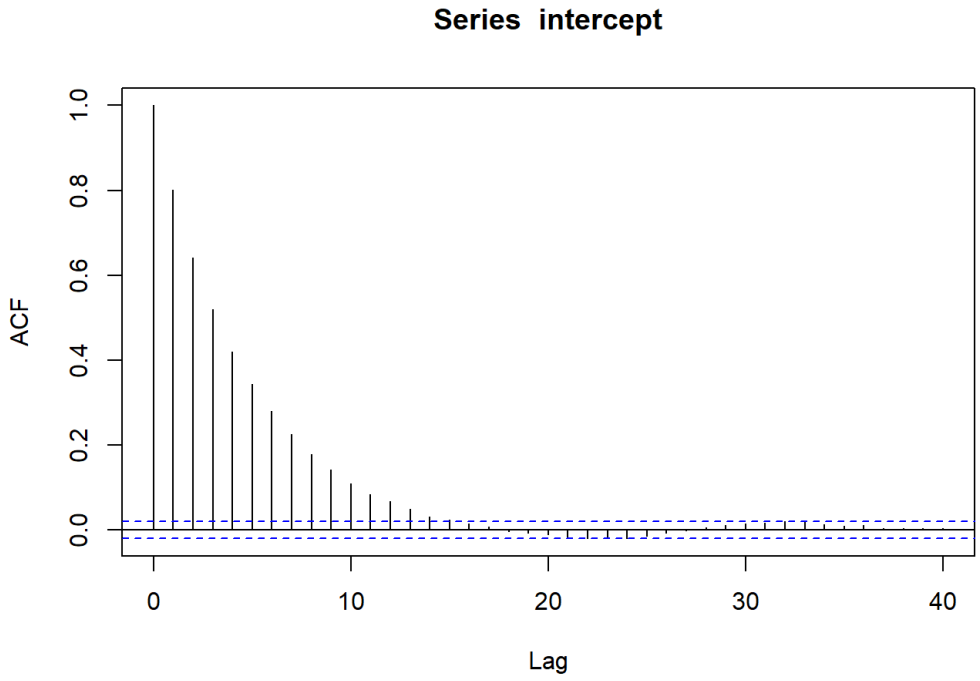
In the fist part of Bayesian part I consider a non informative distribution prior for beta. To compare frequentist and bayesian methods I use the same models made later. Then, I will choose the best one and try to change some parameters, like **B0** giving more precision ($precision = \phi = \frac{1}{\sigma^2}$, so it is the same of reducing variability) and **beta.start** values. In the first part, I use a uniformative prior distribution of parameters: it assigns equal probabilities to all possibilities. In this case $\hat{\beta}_i \sim N(\beta_i, \phi^{-1})$, where β_i is equal to 0 and ϕ to 0.0001 (that means σ^2 is setted to 10.000).

- Model with **agecat** with MCMC

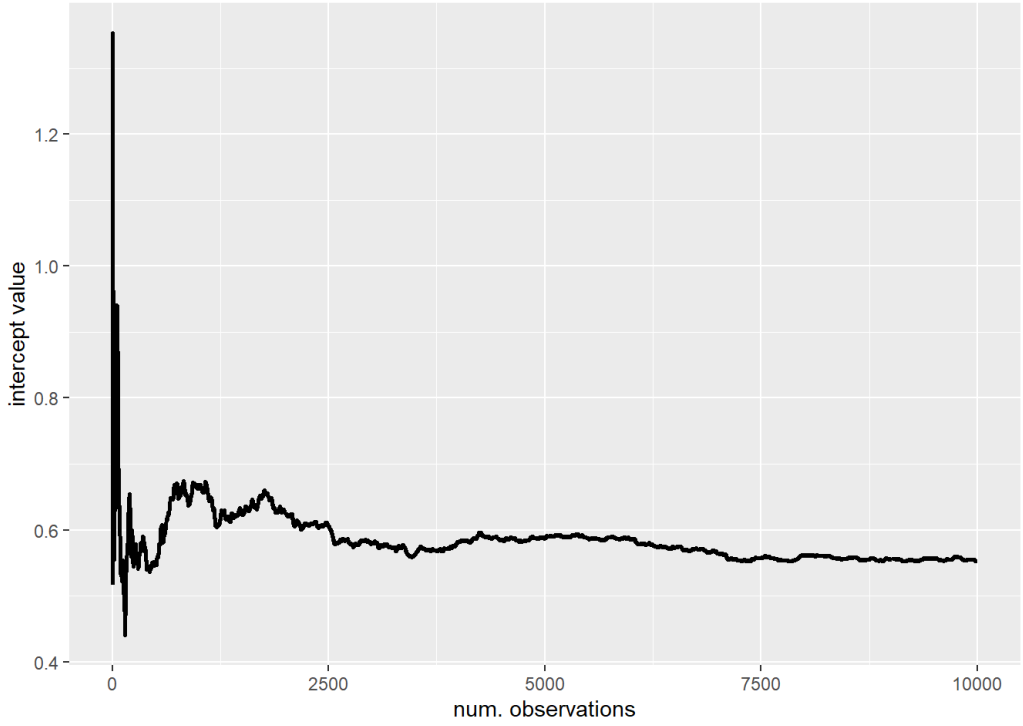
```
bA <- MCMClogit(low ~ age, b_train, mcmc=10000,B0=0.0001,marginal.likelihood="Laplace")



summary(bA)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean      SD  Naive SE Time-series SE
## (Intercept)  0.55216 0.75988 0.0075988      0.0228727
## age         -0.05786 0.03266 0.0003266      0.0009775
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%    97.5%
## (Intercept) -0.9691  0.03102  0.57337  1.07966 2.013482
## age         -0.1218 -0.08103 -0.05802 -0.03508 0.005688
```

```
intercept<-bA[,1]
acf(intercept)
```

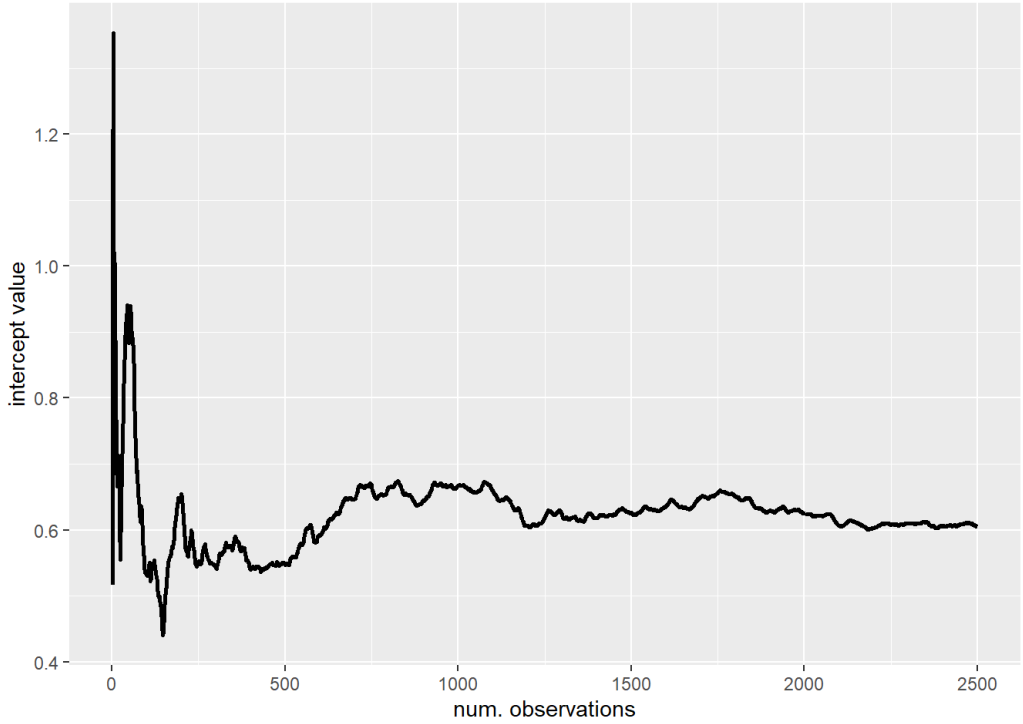



```
y<-cumsum(intercept)/c(1:10000)
y<-data.frame(c=c(1:10000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("intercept value")
```



The autocorrelation plot of **intercept** reveals there is autocorrelation till 15^{th} lag. While the cumulate plot reveals after 2000 iterations the algorithm converges.

The cumulate plot for the first 2500 values shows the **burnin** value could be setted with value of 1000.

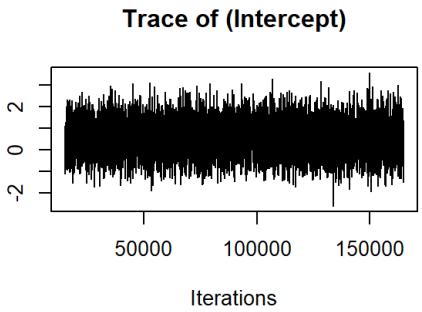


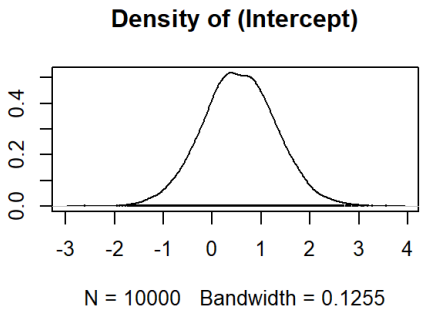
Set new model using **thin** value = 15 and **burnin** value = 1000.

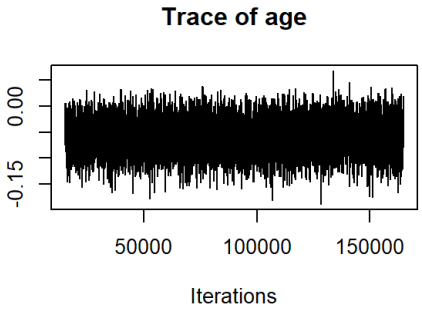
```
bA <- MCMClogit(low ~ age, b_train,burnin=1000*15,mcmc=10000*15, thin =15,
                        B0=0.0001,marginal.likelihood="Laplace")
```

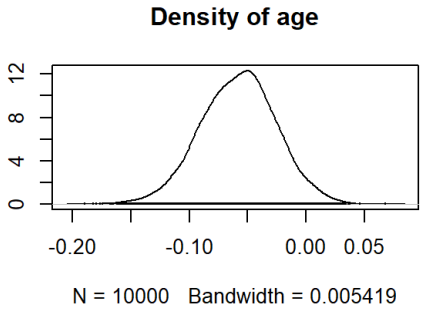
```
##
## Iterations = 15001:164986
## Thinning interval = 15
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean      SD  Naive SE Time-series SE
## (Intercept)  0.54987 0.74882 0.0074882      0.0076225
## age         -0.05802 0.03225 0.0003225      0.0003305
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%    97.5%
## (Intercept) -0.9173  0.05016  0.54505  1.05090 2.020615
## age         -0.1226 -0.07956 -0.05701 -0.03623 0.004309
```

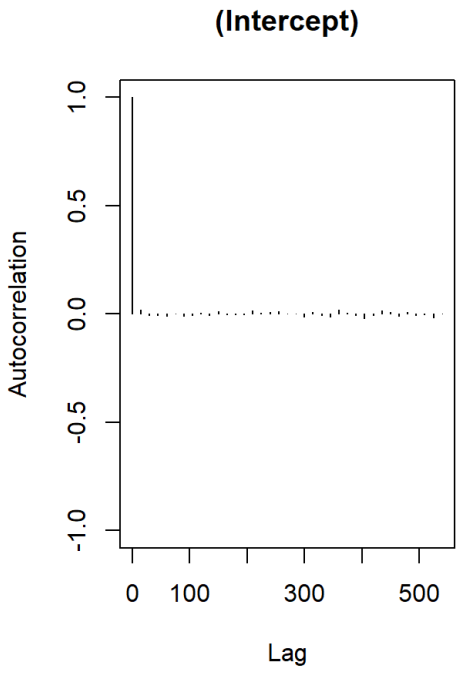
vv

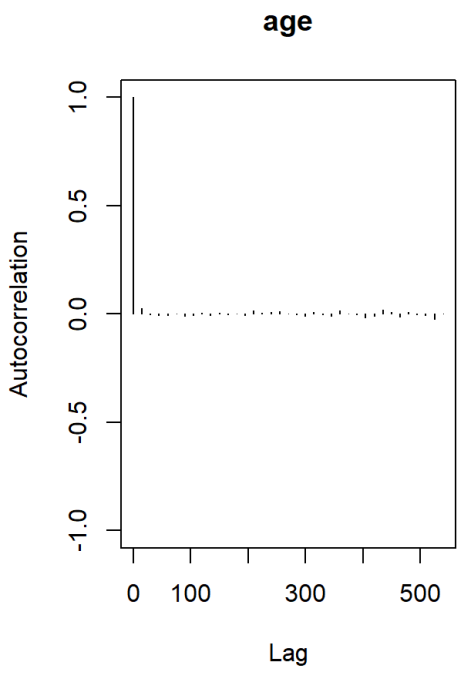

Trace of (Intercept)
-2
0
2
50000
100000
150000
Iterations

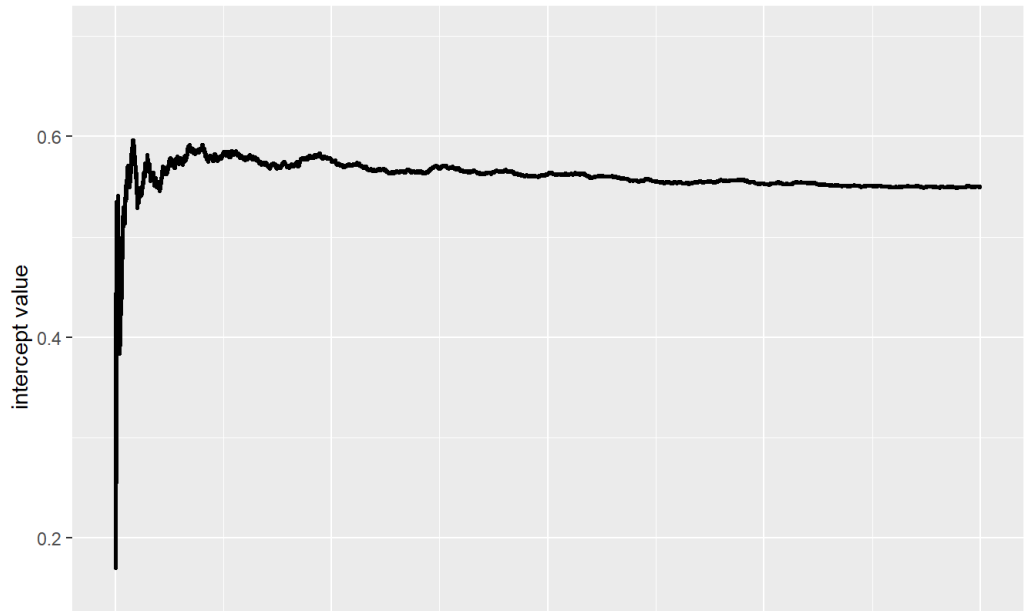

Density of (Intercept)
0.0
0.2
0.4
-3
-2
-1
0
1
2
3
4
N = 10000 Bandwidth = 0.1255

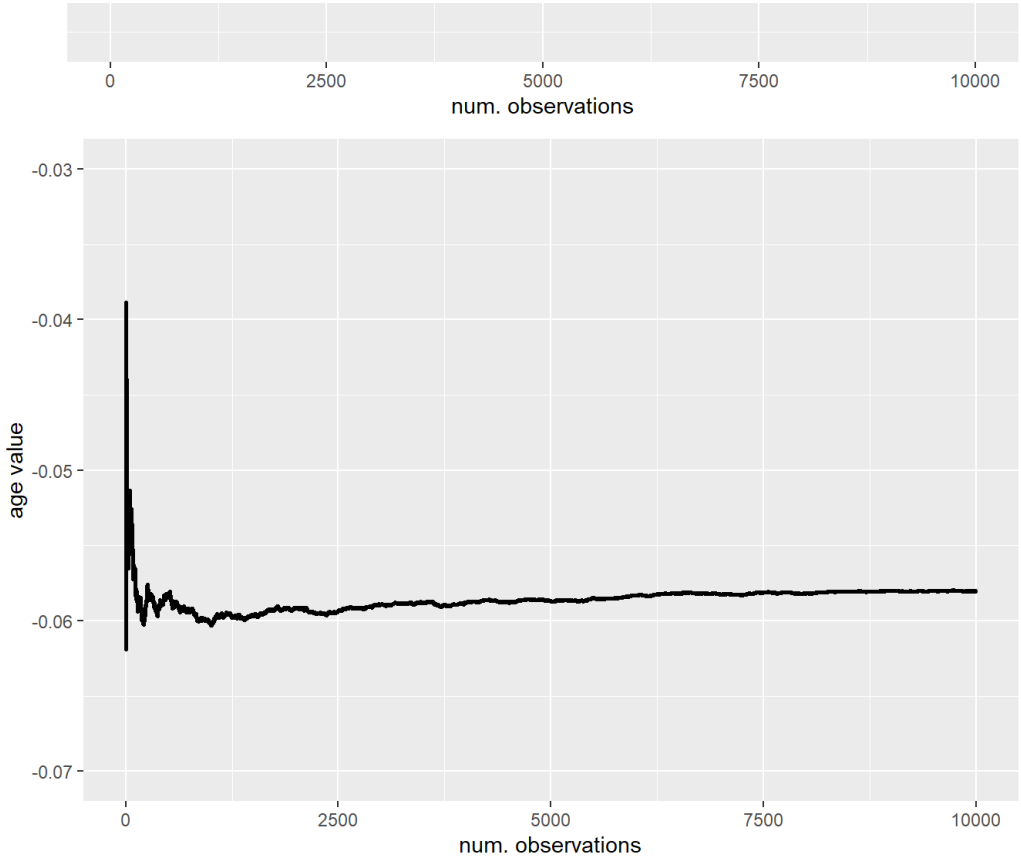

Trace of age
-0.15
0.00
50000
100000
150000
Iterations


Density of age
0
4
8
12
-0.20
-0.10
0.00
0.05
N = 10000 Bandwidth = 0.005419


(Intercept)
Autocorrelation
-1.0
-0.5
0.0
0.5
1.0
0
100
300
500
Lag

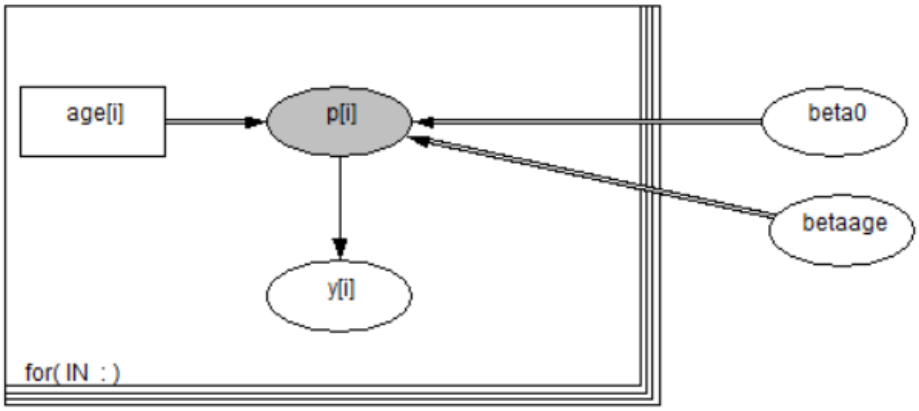

age
Autocorrelation
-1.0
-0.5
0.0
0.5
1.0
0
100
300
500
Lag


intercept value
0.2
0.4
0.6



Trace plots is a time series plot that shows the realizations of the Markov chain at each iteration against the iteration numbers. Autocorrelation reveal no correlation between lags. Now I will run the same model in OpenBugs, it will be helpful later to choose the best model comparing DIC values. By the way, when I use OpenBugs method I will set **n.iter** equals to 10.000 because the method is not fast and needs more time to generate samples. Besides, to save time, I decide to set the initial values in which algorithm starts sampling using frequentist parameters values of the same models.

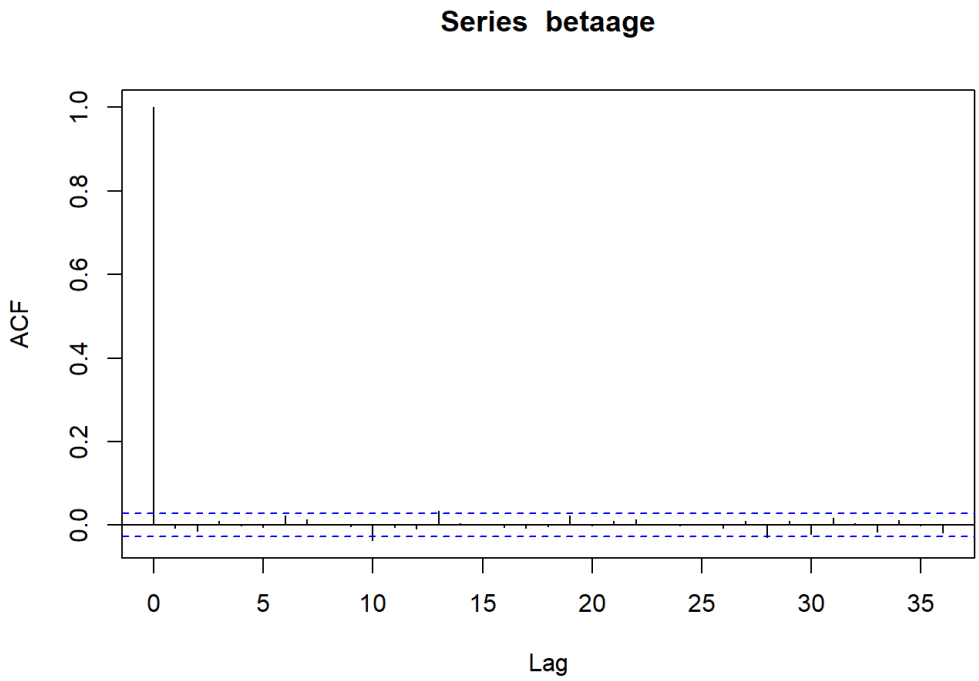
name: p[i] type: logical link: logit
value:



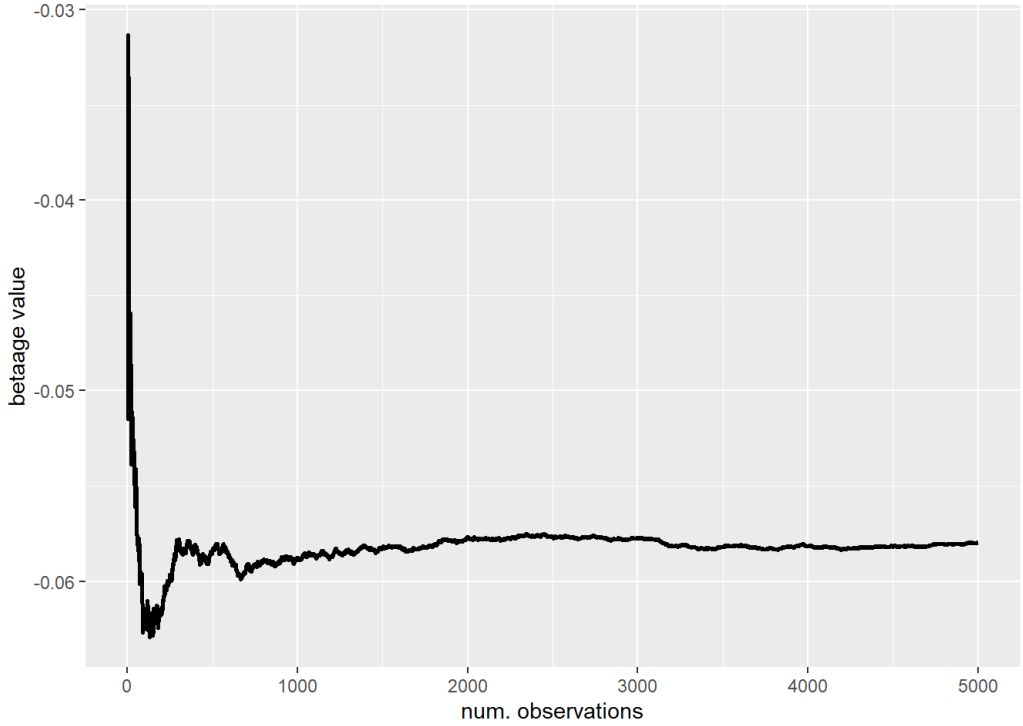
```
y <- b_train$low
n <- length(y)
age <-b_train$age


mlg <- function(){
  for (i in 1:n) {
    y[i] ~ dbern(pi[i])
    logit(pi[i]) <- intercept+betaage*age[i]  }
  intercept ~ dnorm(0,0.0001)
  betaage ~ dnorm(0,0.0001)
}

mlgdata <- list(n = n,y = y,age = age)

mlginits <- function() {
  list(intercept=0.5,betaage=0)
}
mlgoutA <- bugs(data=mlgdata,inits=mlginits,parameters.to.save=c("intercept","betaage"),model.file=mlg,
               n.chains = 1,n.iter = 10000)
mlgoutA
```

```
## Inference for Bugs model at "C:/Users/franc/AppData/Local/Temp/RtmpWafK5Z/model395421bc1b05.txt",
## Current: 1 chains, each with 10000 iterations (first 5000 discarded)
## Cumulative: n.sims = 5000 iterations saved
##            mean  sd  2.5%   25%   50%   75% 97.5%
## intercept   0.5 0.8  -1.0   0.0   0.5   1.1   2.1
## betaage    -0.1 0.0  -0.1  -0.1  -0.1   0.0   0.0
## deviance  223.6 2.0 221.6 222.2 223.0 224.4 229.2
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 2.0 and DIC = 225.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
betaage <- mlgoutA$sims.list$betaage
acf(betaage)
```




```
y<-cumsum(betaage)/c(1:5000)
y<-data.frame(c=c(1:5000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("betaage value")
```



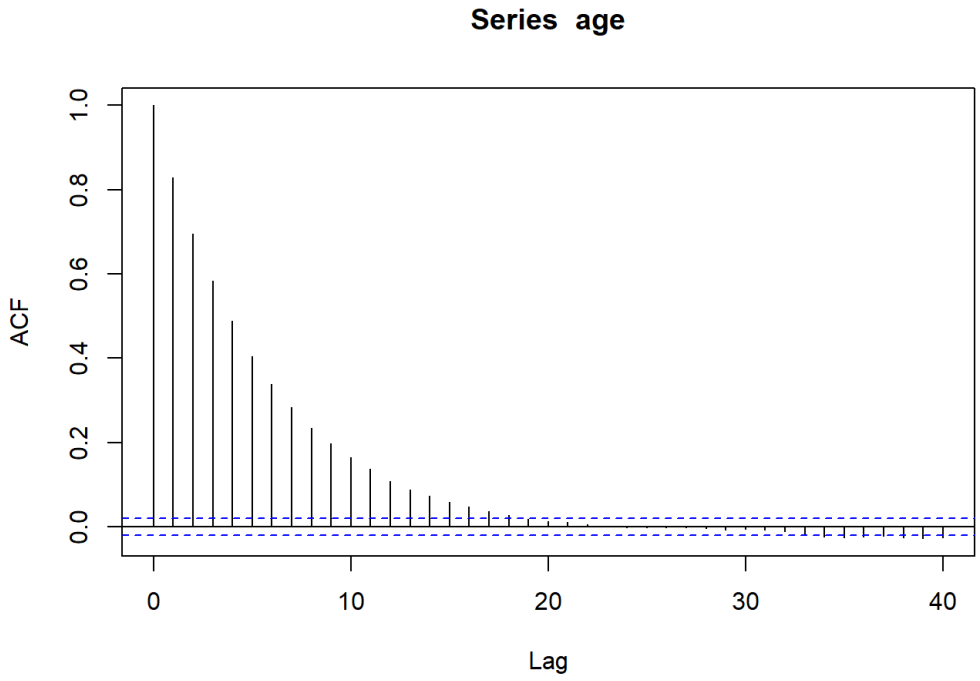
As you can see, the values of MCMC model are the same with the two methods.

- Model with **agecat** + **lwt** with MCMC

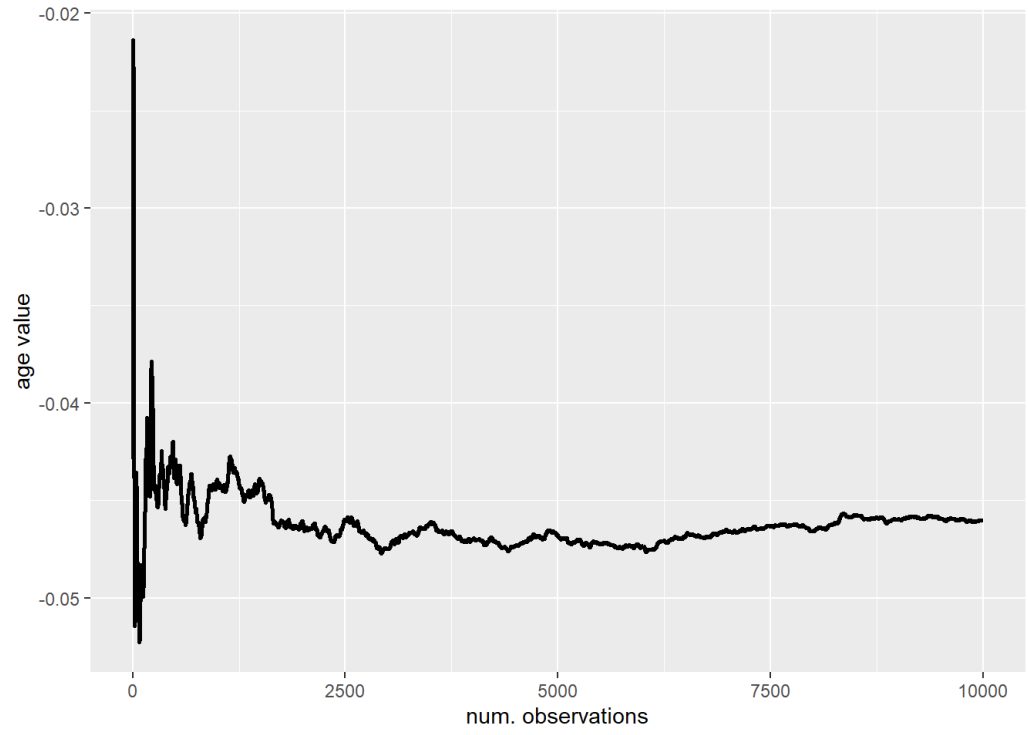
```
bB <- MCMClogit(low ~ age+lwt, b_train,mcmc=10000,B0=0.0001,marginal.likelihood="Laplace")



summary(bB)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean      SD  Naive SE Time-series SE
## (Intercept)  1.91206 1.02865 0.0102865      0.0332327
## age         -0.04601 0.03313 0.0003313      0.0011104
## lwt         -0.01290 0.00654 0.0000654      0.0002213
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%      97.5%
## (Intercept) -0.0612  1.20194  1.89905  2.59897  3.9848499
## age         -0.1125 -0.06880 -0.04566 -0.02378  0.0192394
## lwt         -0.0260 -0.01726 -0.01280 -0.00836 -0.0007851
```

```
age<-bB[,2]
acf(age)
```



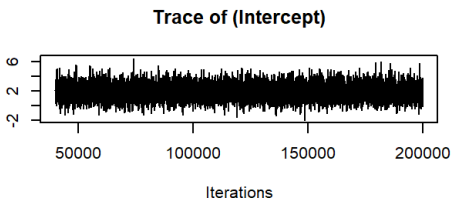

```
y<-cumsum(age)/c(1:10000)
y<-data.frame(c=c(1:10000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("age value")
```

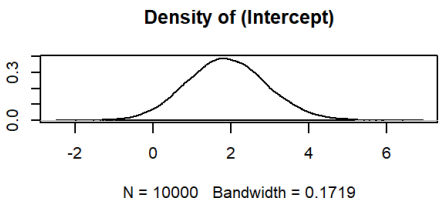


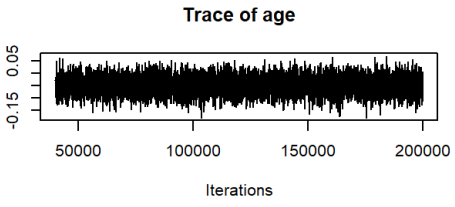
Autocorrelation plot reveals there is correlation till 16^{th} lag. While the algorithm converges after more less 2500 iterations.

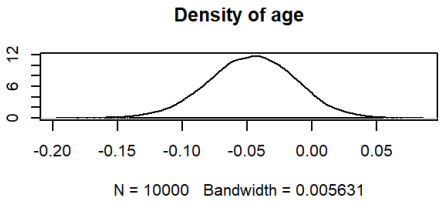
```
bB <- MCMClogit(low ~ age+lwt, b_train,burnin=2500*16,mcmc=10000*16, thin =16,
                        B0=0.0001,marginal.likelihood="Laplace")
```

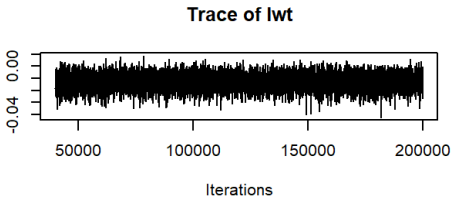
```
##
## Iterations = 40001:199985
## Thinning interval = 16
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  1.89435 1.024814 1.025e-02      0.0110101
## age         -0.04647 0.033518 3.352e-04      0.0003596
## lwt         -0.01264 0.006308 6.308e-05      0.0000685
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%      50%       75%      97.5%
## (Intercept) -0.06908  1.19837  1.86977  2.569184  3.9530315
## age         -0.11390 -0.06871 -0.04588 -0.023522  0.0180245
## lwt         -0.02540 -0.01675 -0.01246 -0.008252 -0.0009251
```

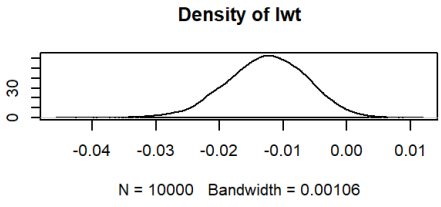

Trace of (Intercept)
-2
2
6
50000
100000
150000
200000
Iterations

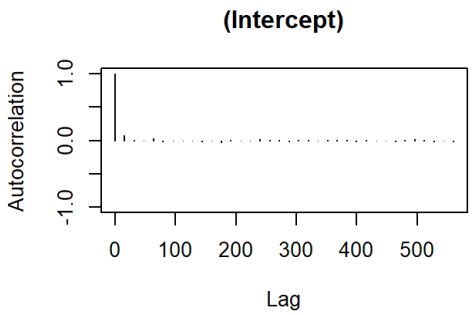

Density of (Intercept)
0.0
0.3
-2
0
2
4
6
N = 10000 Bandwidth = 0.1719

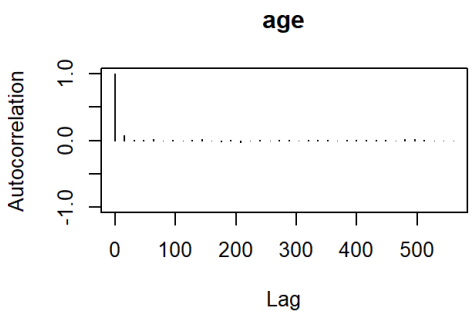

Trace of age
-0.15
0.05
50000
100000
150000
200000
Iterations

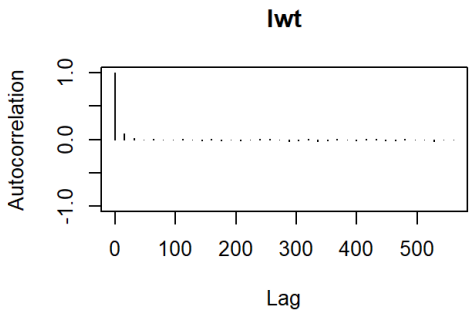

Density of age
0
6
12
-0.20
-0.15
-0.10
-0.05
0.00
0.05
N = 10000 Bandwidth = 0.005631

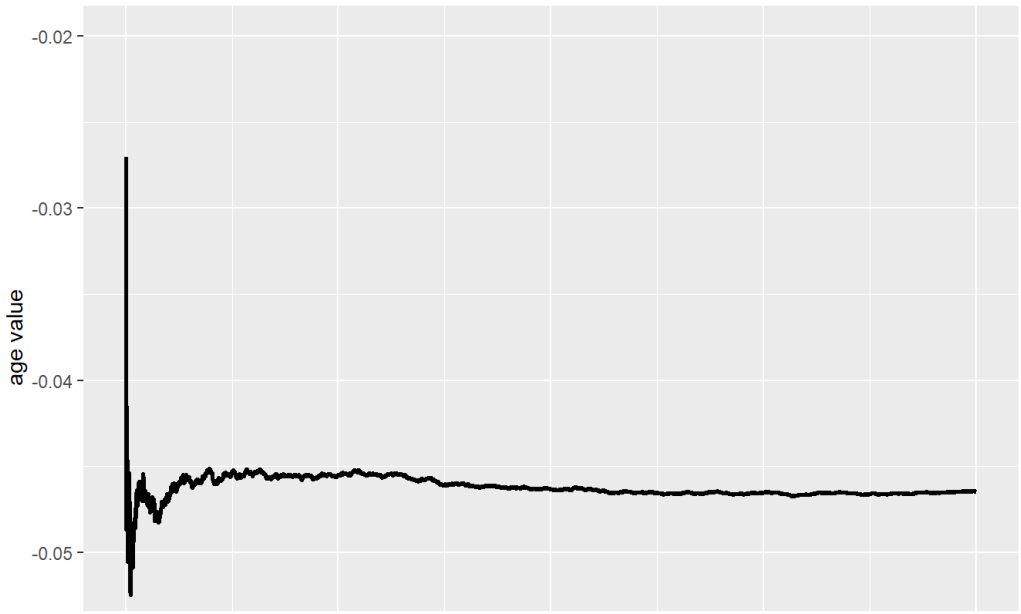

Trace of lwt
-0.04
0.00
50000
100000
150000
200000
Iterations

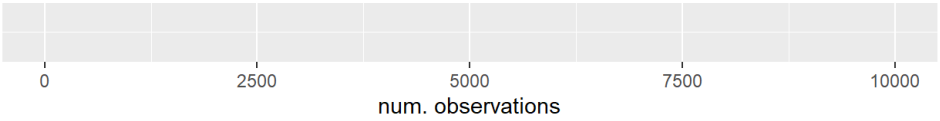

Density of lwt
0
30
-0.04
-0.03
-0.02
-0.01
0.00
0.01
N = 10000 Bandwidth = 0.00106

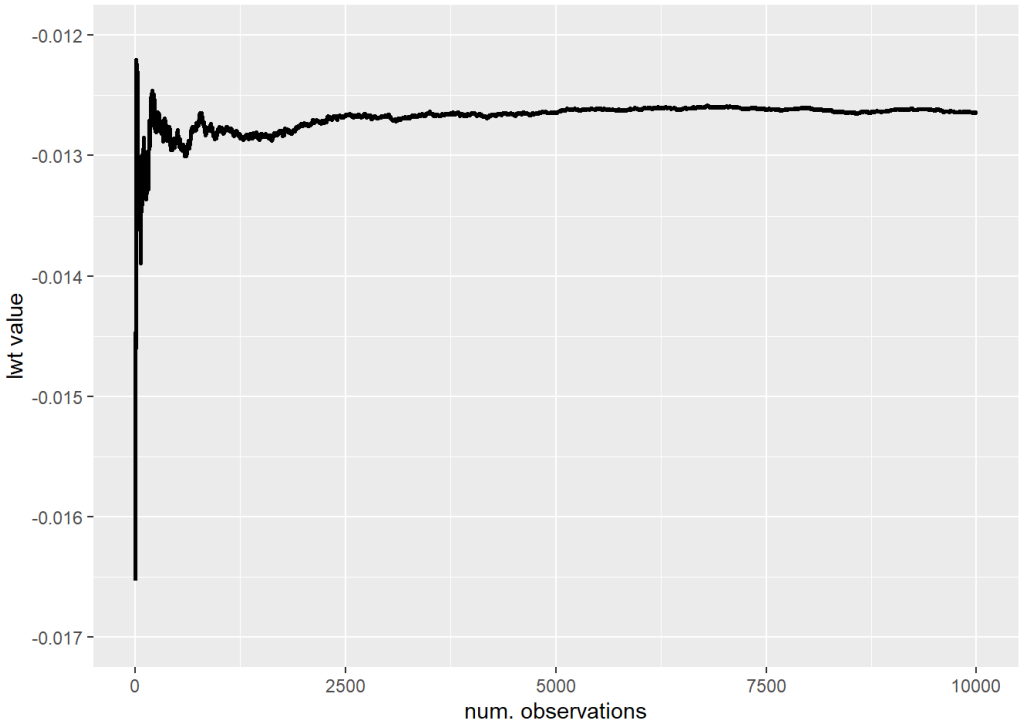

(Intercept)
Autocorrelation
-1.0
0.0
1.0
0
100
200
300
400
500
Lag


age
Autocorrelation
-1.0
0.0
1.0
0
100
200
300
400
500
Lag


lwt
Autocorrelation
-1.0
0.0
1.0
0
100
200
300
400
500
Lag

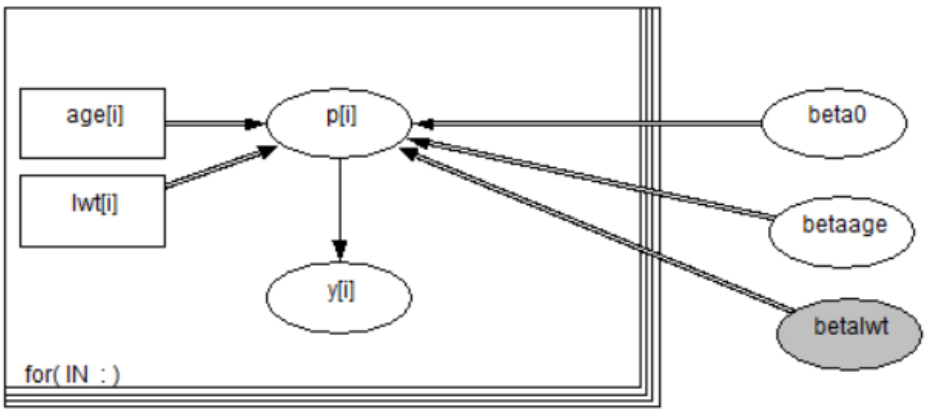

-0.02
-0.03
-0.04
-0.05
age value





Trace plots is good for all parameters. Autocorrelation plot reveals no correlation between lags. And now we go on with OpenBugs

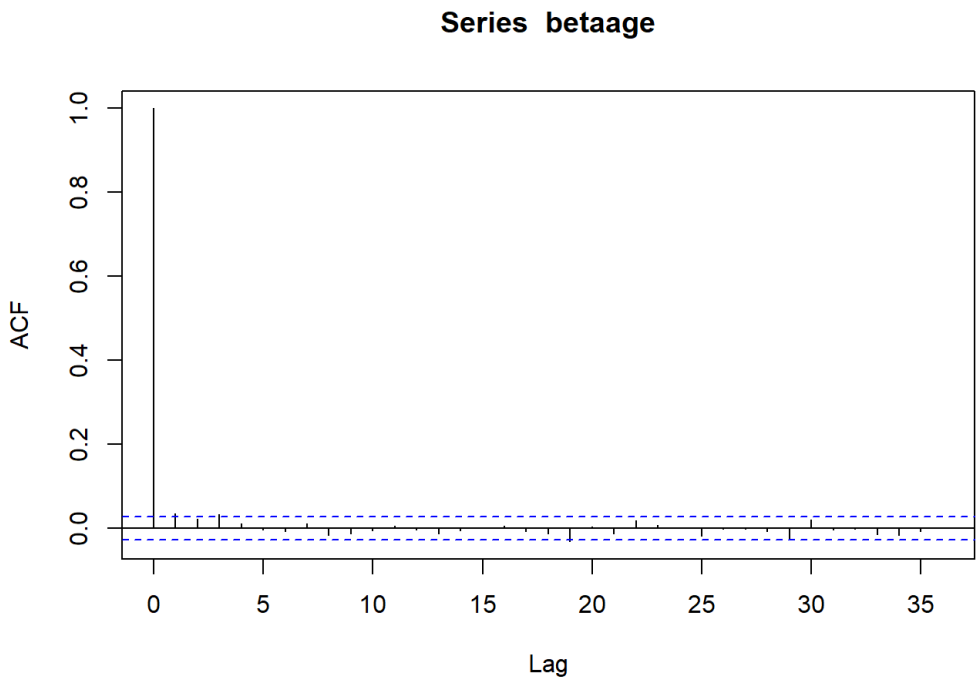
name: betalwt type: stochastic density: dnorm
mean 0.0 precision 1.0E-6 lower bound upper bound



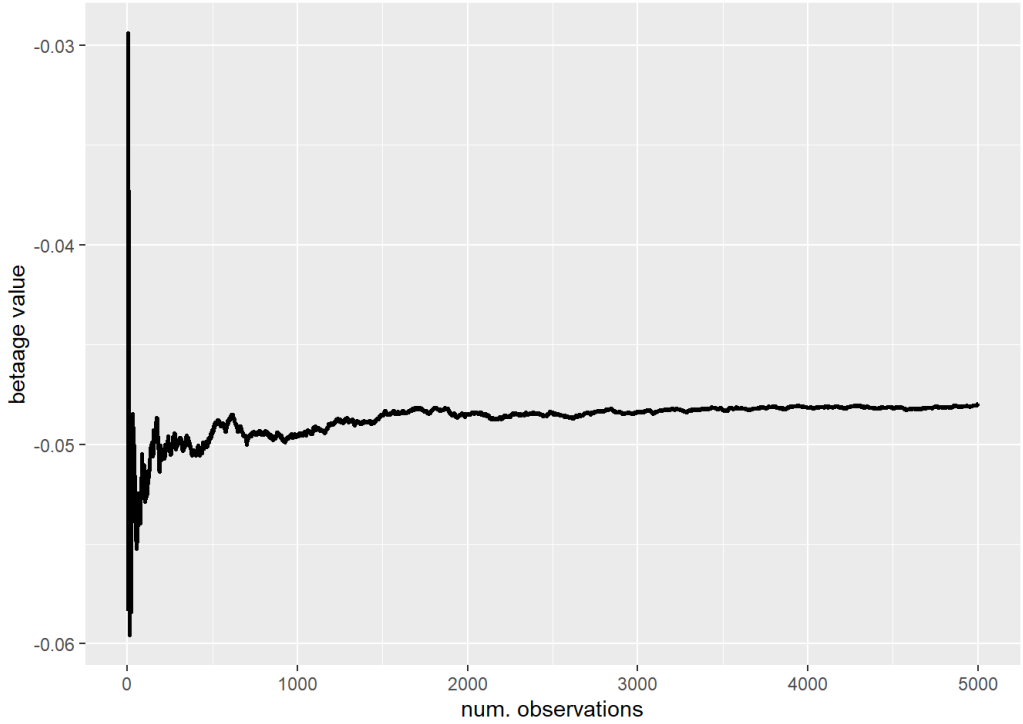
```
y <- b_train$low
n <- length(y)
age <-b_train$age
lwt<-b_train$lwt

mlg <- function(){
  for (i in 1:n) {
    y[i] ~ dbern(pi[i])
    logit(pi[i]) <- intercept+betaage*age[i]+betalwt*lwt[i] }
  intercept ~ dnorm(0,0.0001)
  betaage ~ dnorm(0,0.0001)
  betalwt ~ dnorm(0,0.0001)
}

mlgdata <- list(n = n,y = y,age = age,lwt=lwt)

mlginits <- function() {
  list(intercept=1.9,betaage=0,betalwt=0)
}
mlgoutB <- bugs(data=mlgdata,inits=mlginits,parameters.to.save=c("intercept","betaage","betalwt"),model.file=mlg,
               n.chains = 1,n.iter = 10000)
mlgoutB
```

```
## Inference for Bugs model at "C:/Users/franc/AppData/Local/Temp/RtmpWafK5Z/model39545f2666d3.txt",
## Current: 1 chains, each with 10000 iterations (first 5000 discarded)
## Cumulative: n.sims = 5000 iterations saved
##            mean  sd  2.5%   25%   50%   75% 97.5%
## intercept   1.9 1.0   0.0   1.3   1.9   2.6   4.0
## betaage     0.0 0.0  -0.1  -0.1   0.0   0.0   0.0
## betalwt     0.0 0.0   0.0   0.0   0.0   0.0   0.0
## deviance  220.4 2.6 217.6 218.5 219.7 221.5 227.7
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 3.0 and DIC = 223.5
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
betaage <- mlgoutB$sims.list$betaage
acf(betaage)
```




```
y<-cumsum(betaage)/c(1:5000)
y<-data.frame(c=c(1:5000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("betaage value")
```

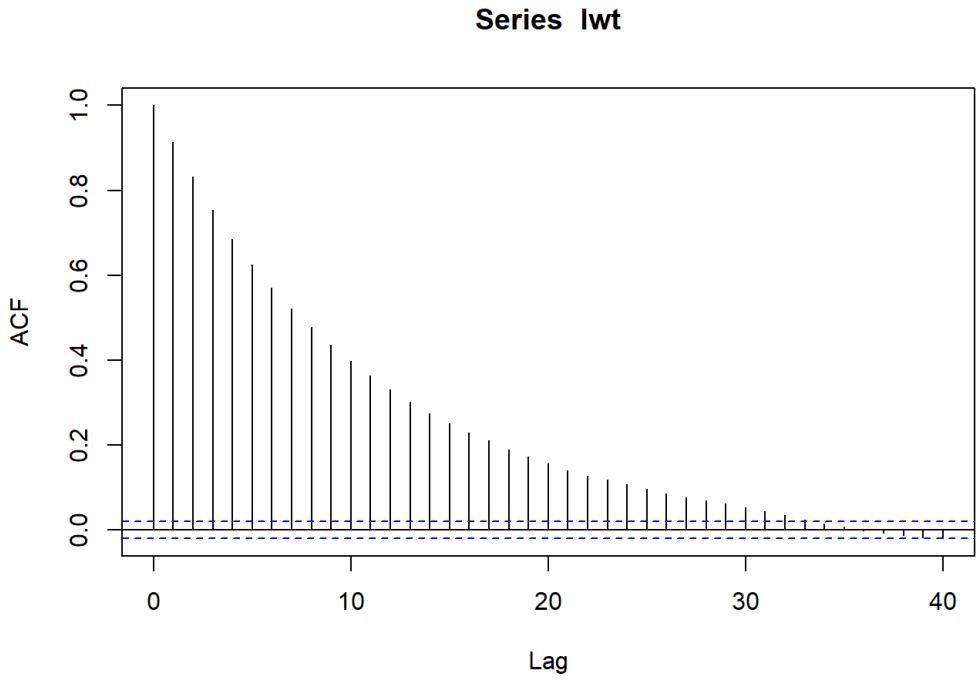


- Model with **best frequentist** model parameters with MCMC

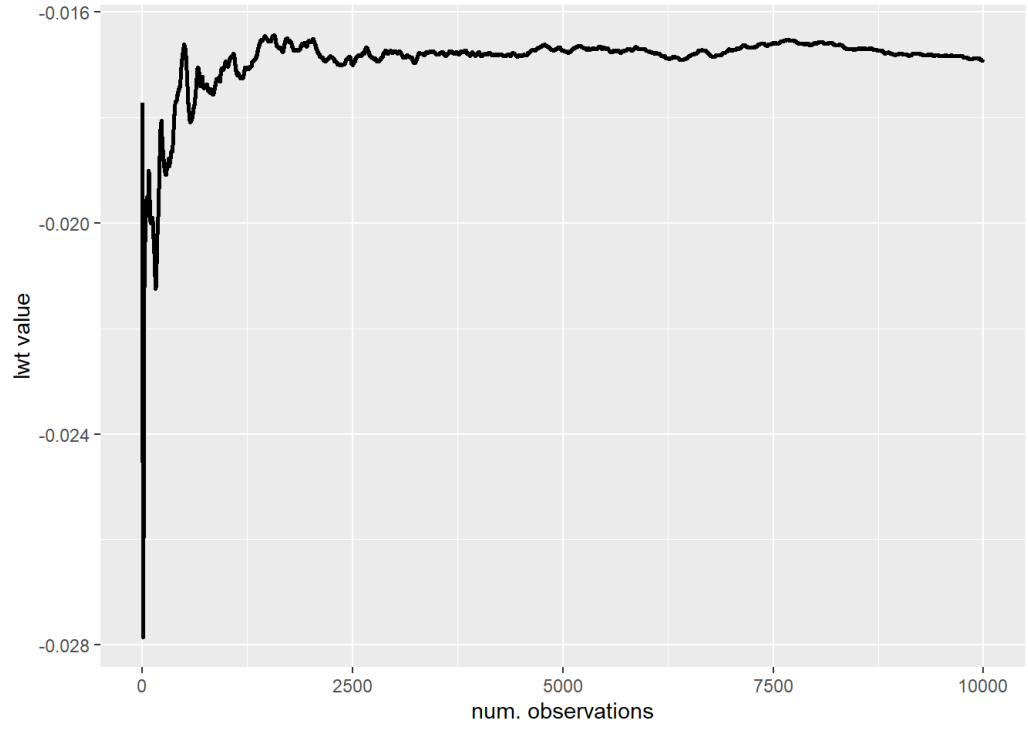
```
bM <- MCMClogit(low ~ lwt+ht+smoke+ui+race, b_train,mcmc=10000,B0=0.0001,marginal.likelihood="Laplace")



summary(bM)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  0.01196 0.958337 9.583e-03      0.0458586
## lwt         -0.01690 0.006824 6.824e-05      0.0003106
## ht1          1.84186 0.676566 6.766e-03      0.0309133
## smoke1       1.16872 0.413380 4.134e-03      0.0198533
## ui1          0.85881 0.449474 4.495e-03      0.0212573
## race2        1.43289 0.540631 5.406e-03      0.0252023
## race3        0.92836 0.451940 4.519e-03      0.0217178
##
## 2. Quantiles for each variable:
##
##                 2.5%     25%      50%      75%     97.5%
## (Intercept) -1.86869 -0.6354  0.01137  0.66252  1.936115
## lwt         -0.03055 -0.0213 -0.01679 -0.01227 -0.003987
## ht1          0.57394  1.3735  1.81120  2.27113  3.243039
## smoke1       0.32099  0.8902  1.17475  1.42892  1.998890
## ui1          0.01822  0.5517  0.85152  1.15395  1.805424
## race2        0.36788  1.0904  1.44067  1.78943  2.482954
## race3        0.07065  0.6288  0.92423  1.23121  1.840905
```

```
lwt<-bM[,2]
acf(lwt)
```



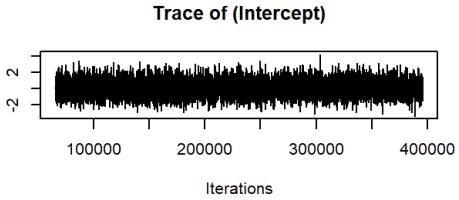

```
y<-cumsum(lwt)/c(1:10000)
y<-data.frame(c=c(1:10000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("lwt value")
```

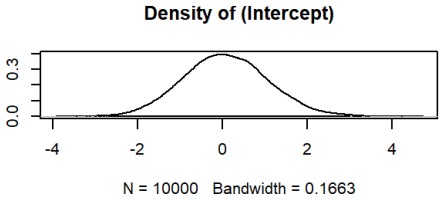


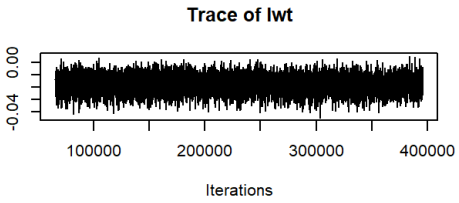
The ACF plot reveals there is correlation till 33^{th} lag and parameter converges after 2000 observations. I will deleting it setting **thin** value.

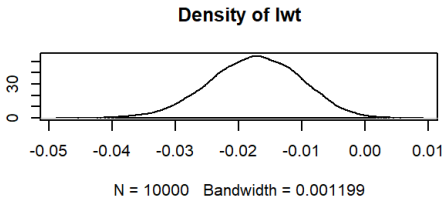
```
bM <- MCMClogit(low ~ lwt+ht+smoke+ui+race, b_train,burnin=2000*33,mcmc=10000*33, thin =33,
                        B0=0.0001,marginal.likelihood="Laplace")
```

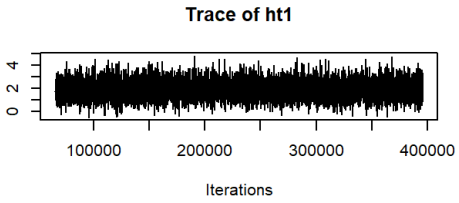
```
##
## Iterations = 66001:395968
## Thinning interval = 33
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  0.07060 0.989640 9.896e-03      1.065e-02
## lwt         -0.01743 0.007136 7.136e-05      7.625e-05
## ht1          1.92775 0.725057 7.251e-03      7.830e-03
## smoke1       1.18481 0.415914 4.159e-03      4.410e-03
## ui1          0.89582 0.465405 4.654e-03      4.909e-03
## race2        1.40101 0.541571 5.416e-03      5.773e-03
## race3        0.91801 0.452041 4.520e-03      4.829e-03
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%     97.5%
## (Intercept) -1.81930 -0.60203  0.04925  0.72480  2.062679
## lwt         -0.03188 -0.02223 -0.01725 -0.01242 -0.004051
## ht1          0.57440  1.43893  1.90643  2.40135  3.399136
## smoke1       0.38290  0.90047  1.17909  1.46429  2.005434
## ui1         -0.01577  0.58657  0.89451  1.21312  1.806197
## race2        0.33852  1.03646  1.39952  1.76161  2.475103
## race3        0.02250  0.61354  0.91898  1.21108  1.805597
```

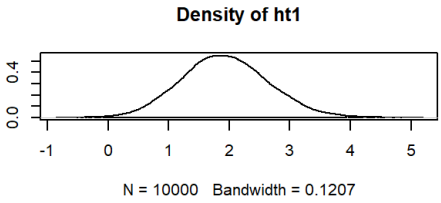

Trace of (Intercept)
Iterations

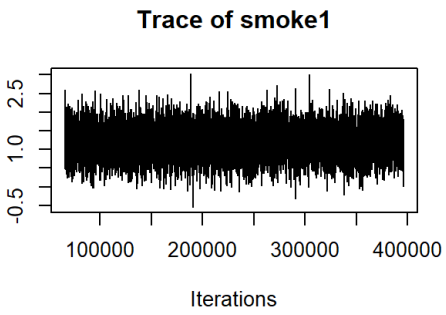

Density of (Intercept)
N = 10000 Bandwidth = 0.1663

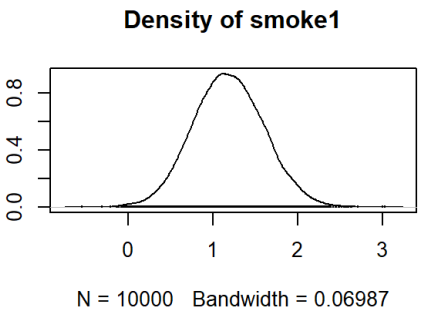

Trace of lwt
Iterations

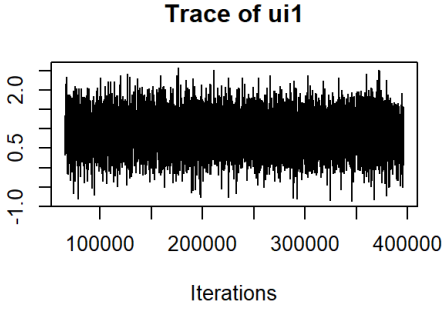

Density of lwt
N = 10000 Bandwidth = 0.001199

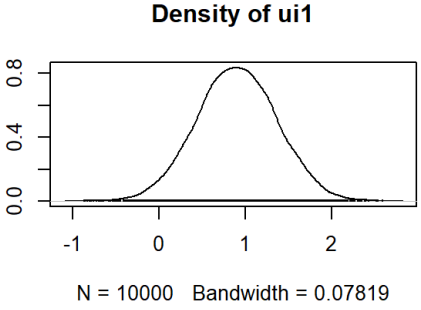

Trace of ht1
Iterations

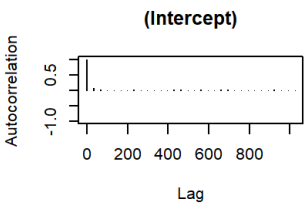

Density of ht1
N = 10000 Bandwidth = 0.1207

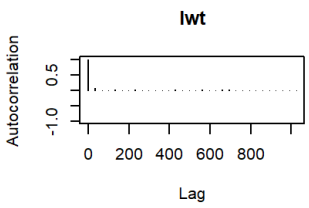

Trace of smoke1
Iterations

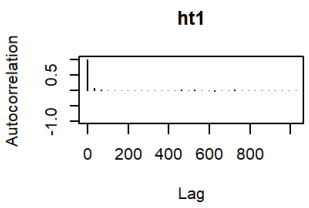

Density of smoke1
N = 10000 Bandwidth = 0.06987

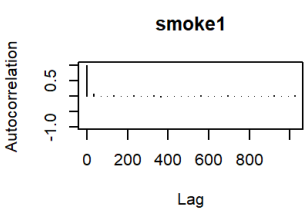

Trace of ui1
Iterations

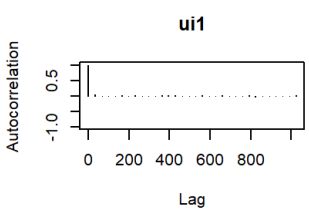

Density of ui1
N = 10000 Bandwidth = 0.07819

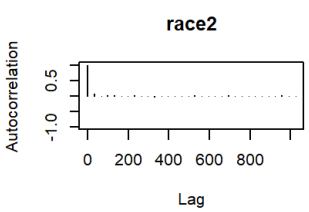

(Intercept)
Autocorrelation
Lag

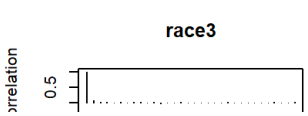

lwt
Autocorrelation
Lag

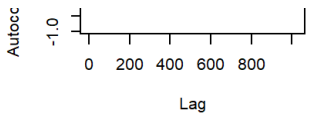

ht1
Autocorrelation
Lag

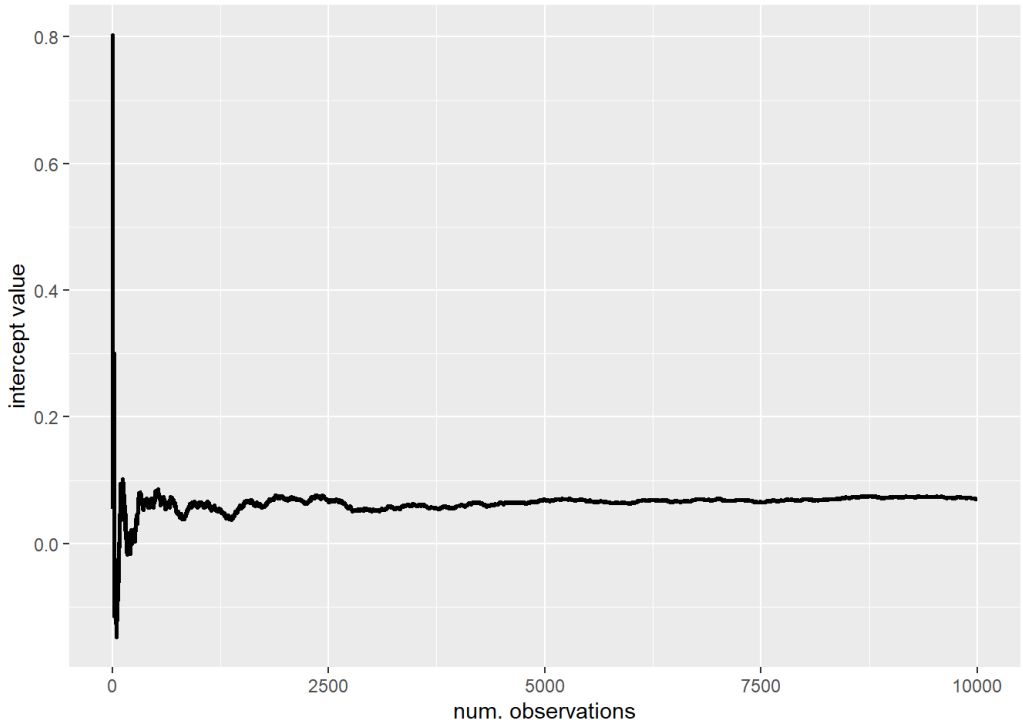

smoke1
Autocorrelation
Lag

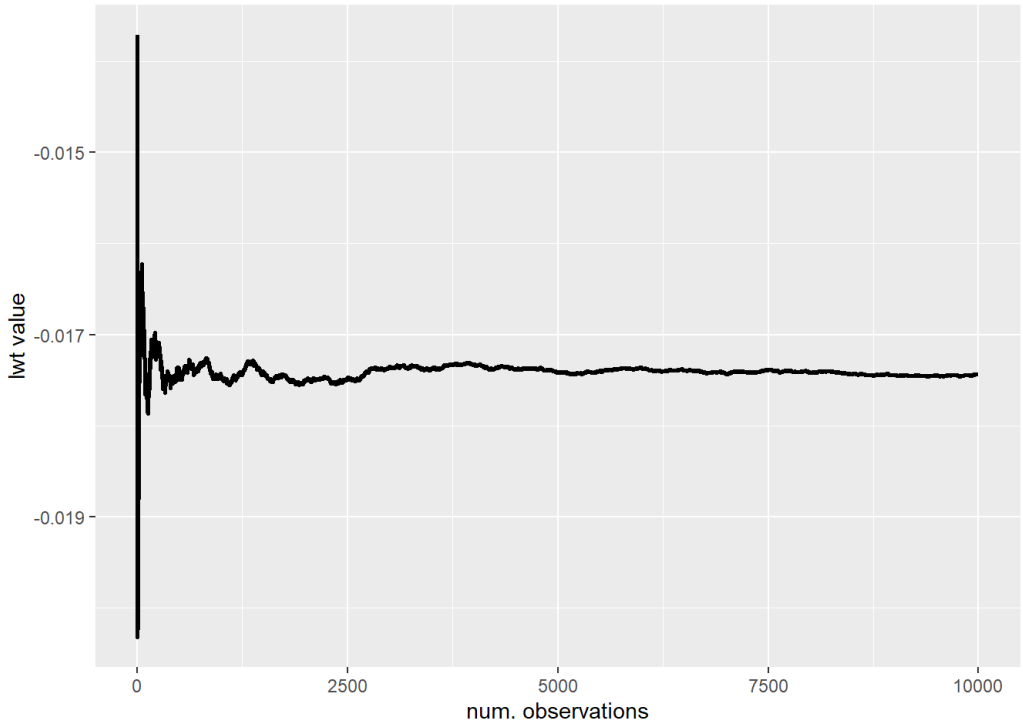

ui1
Autocorrelation
Lag

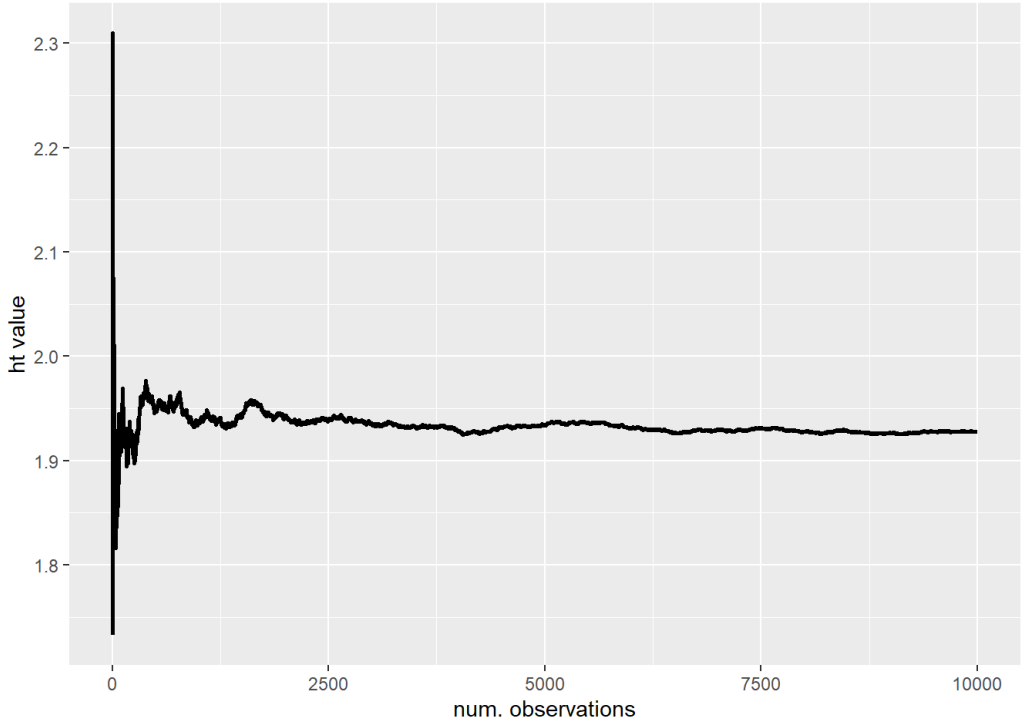

race2
Autocorrelation
Lag

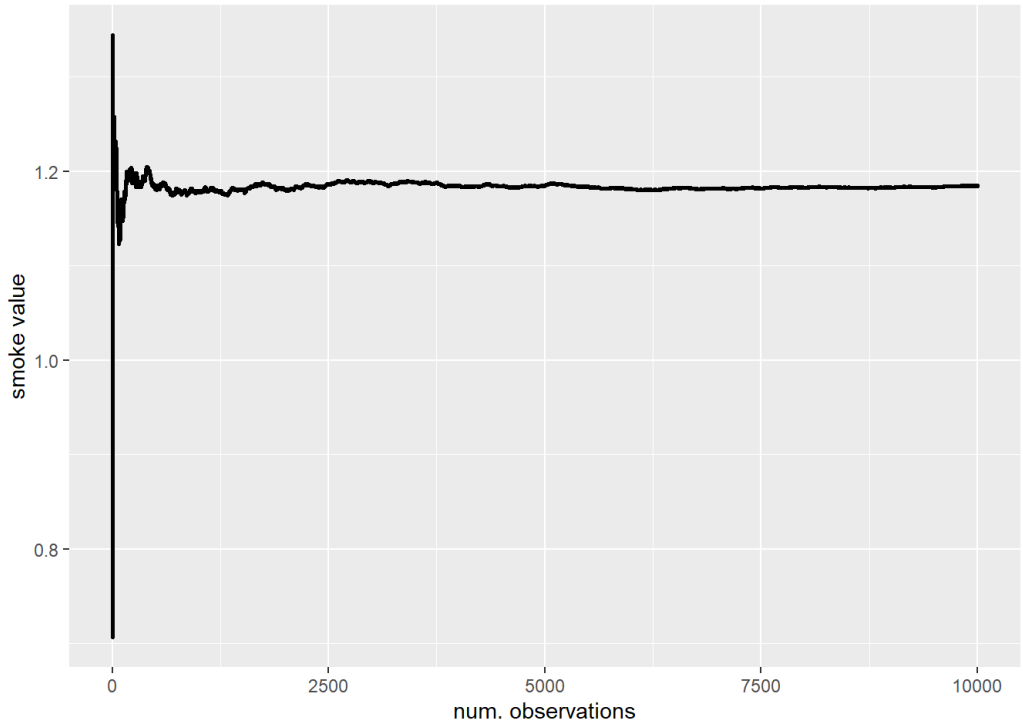

race3

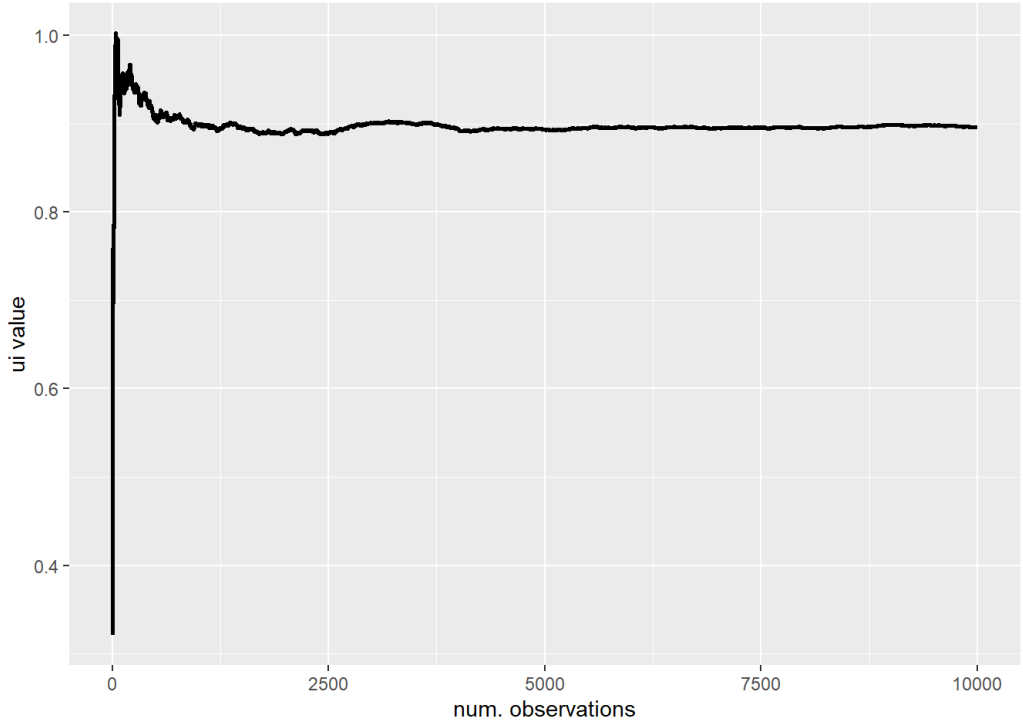

Autocc
-1.0
0
200
400
600
800
Lag

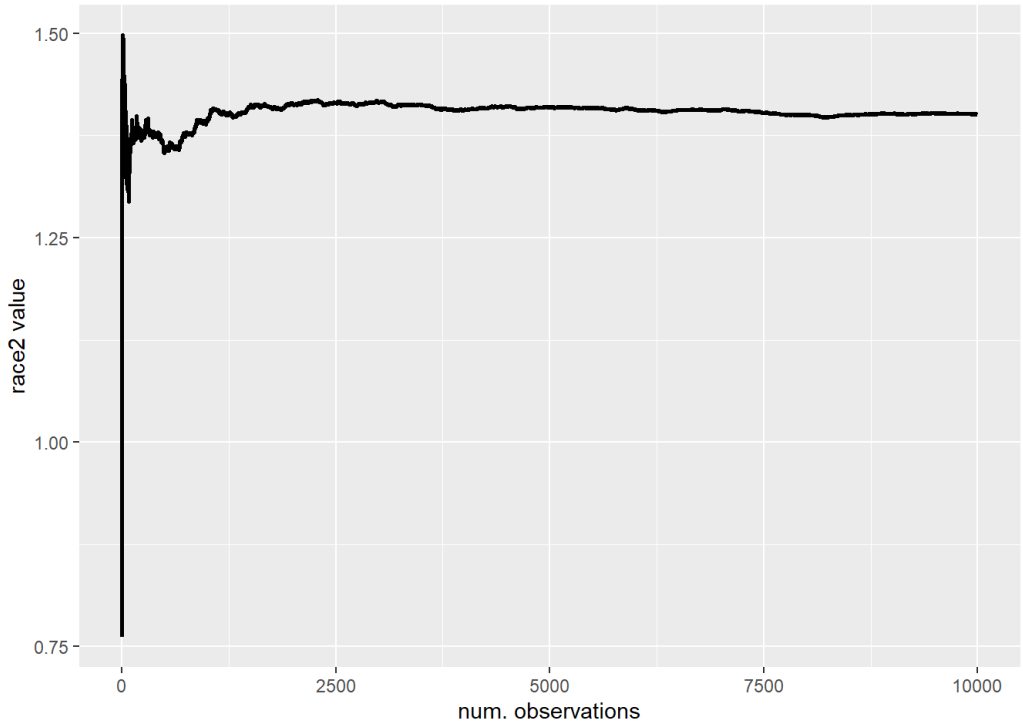

0.8
0.6
0.4
0.2
0.0
intercept value
0
2500
5000
7500
10000
num. observations

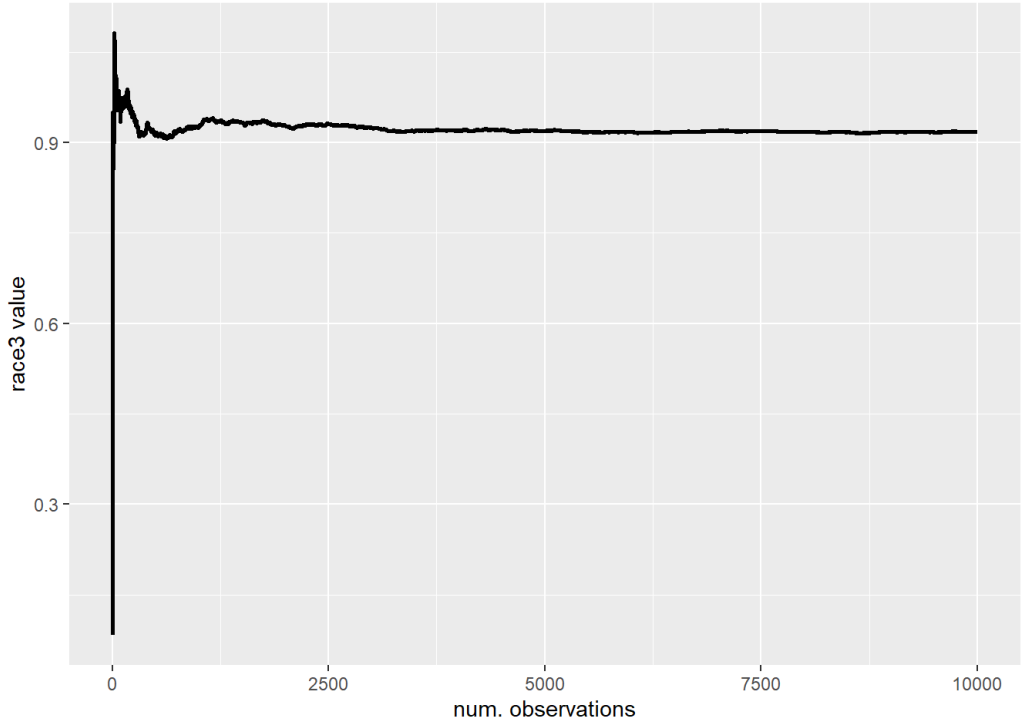

-0.015
-0.017
-0.019
lwt value
0
2500
5000
7500
10000
num. observations


2.3
2.2
2.1
2.0
1.9
1.8
ht value
0
2500
5000
7500
10000
num. observations


1.2
1.0
0.8
smoke value
0
2500
5000
7500
10000
num. observations

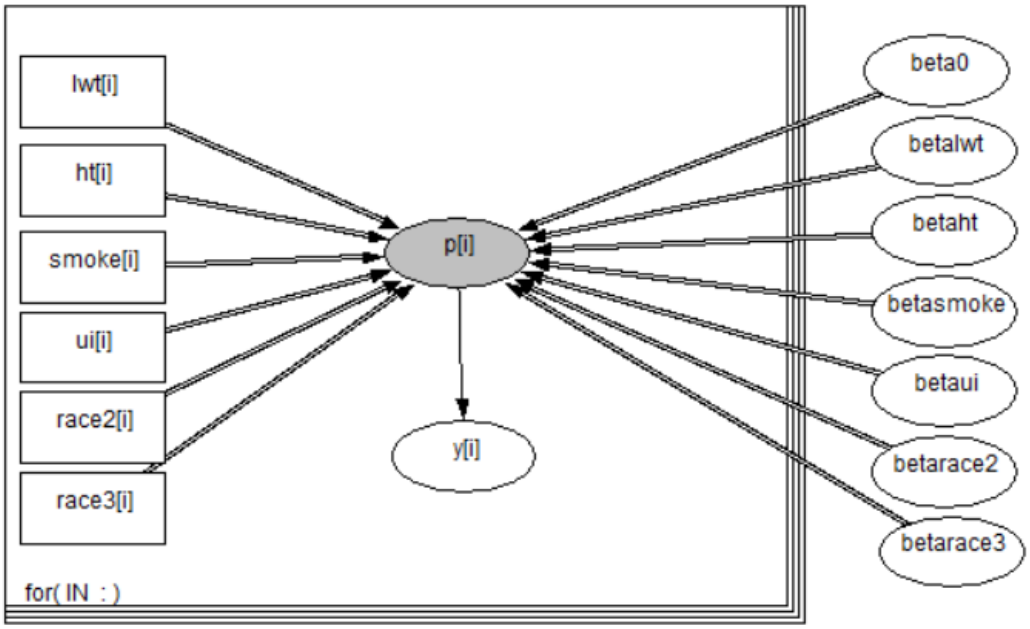

1.0
0.8
0.6
0.4
ui value
0
2500
5000
7500
10000
num. observations


1.50
1.25
1.00
0.75
race2 value
0
2500
5000
7500
10000
num. observations



Making the same with OpenBugs.

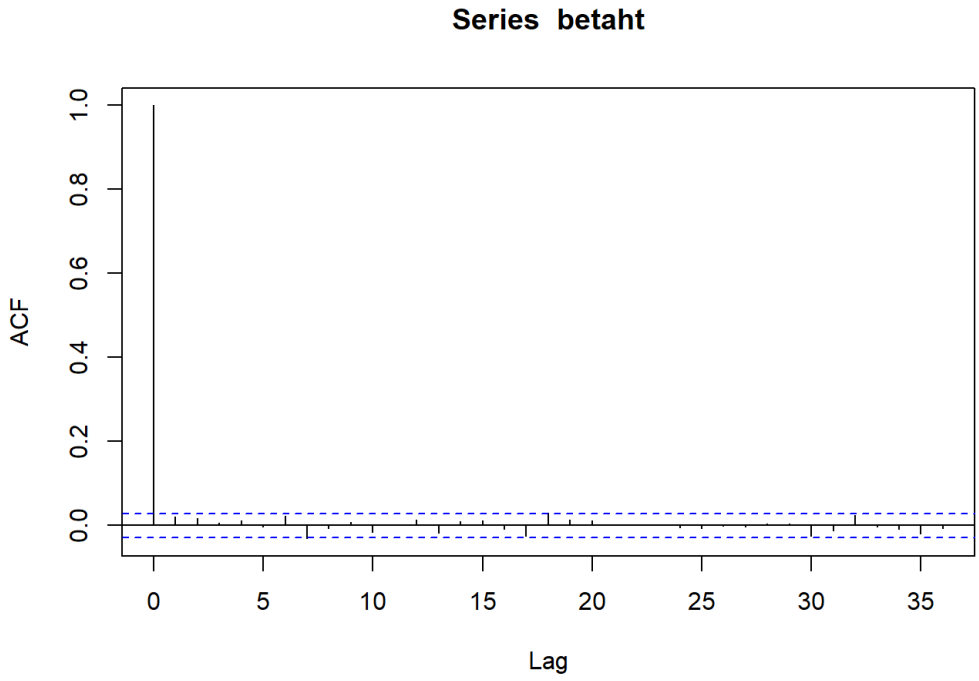
name: p[i] type: logical link: logit
value:



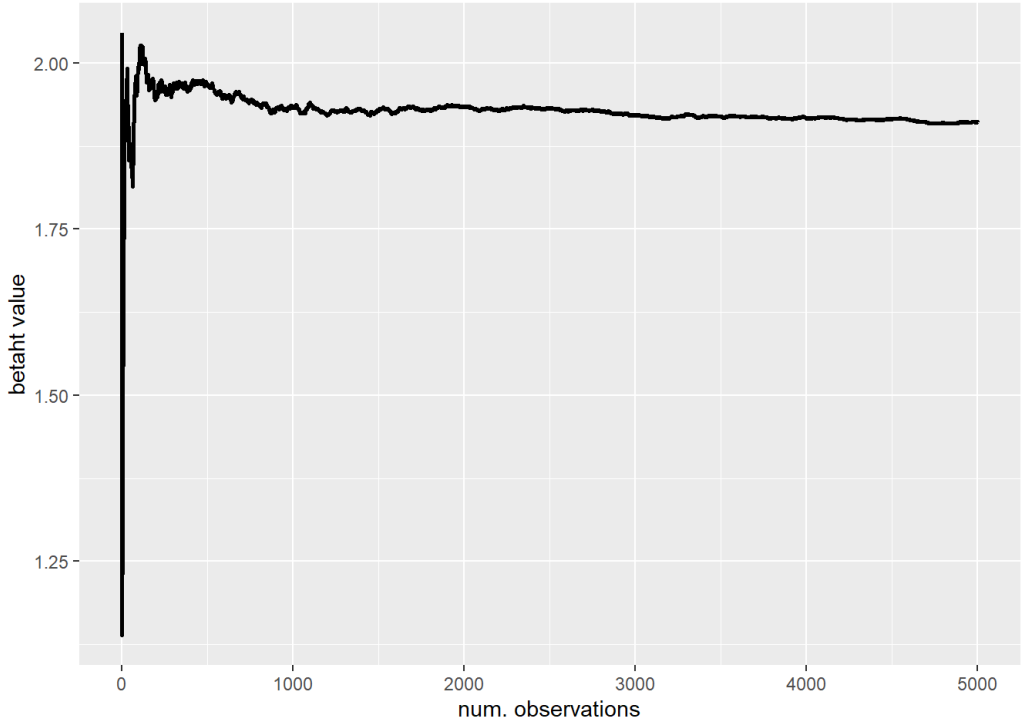
```
y <- b_train$low
n <- length(y)
lwt<-b_train$lwt
ht <- as.numeric(b_train$ht)-1
smoke <- as.numeric(b_train$smoke)-1
ui <- as.numeric(b_train$ui)-1
race <- as.numeric(b_train$race)-1
race2 <- race
race2[race2!=1] <- 0
race3 <- race
race3[race3!=2] <- 0
race3[race3==2] <-1

mlg <- function(){
  for (i in 1:n) {
    y[i] ~ dbern(pi[i])
    logit(pi[i]) <- intercept+betalwt*lwt[i]+betaht*ht[i]+betasmoke*smoke[i]+betaui*ui[i]+betarace2*race2[i]+betarace3*race
3[i] }
  intercept ~ dnorm(0,0.0001)
  betalwt ~ dnorm(0,0.0001)
  betaht ~ dnorm(0,0.0001)
  betasmoke ~ dnorm(0,0.0001)
  betaui ~ dnorm(0,0.0001)
  betarace2 ~ dnorm(0,0.0001)
  betarace3 ~ dnorm(0,0.0001)
}

mlgdata <- list(n = n,y = y,lwt=lwt,ht=ht,smoke=smoke,ui=ui,race2=race2,race3=race3)

mlginits <- function() {
  list(intercept=0,betalwt=0,betaht=1.9,betasmoke=1.1,betaui=0.9,betarace2=1.4,betarace3=0.9)
}
mlgoutC <- bugs(data=mlgdata,inits=mlginits,parameters.to.save=c("intercept","betalwt","betaht","betasmoke","betaui","betar
ace2","betarace3"),model.file=mlg,
               n.chains = 1,n.iter = 10000)
mlgoutC
```

```
## Inference for Bugs model at "C:/Users/franc/AppData/Local/Temp/RtmpWafK5Z/model39541e2833c.txt",
## Current: 1 chains, each with 10000 iterations (first 5000 discarded)
## Cumulative: n.sims = 5000 iterations saved
##            mean  sd  2.5%   25%   50%   75% 97.5%
## intercept   0.0 1.0  -1.9  -0.6   0.0   0.6   2.0
## betalwt     0.0 0.0   0.0   0.0   0.0   0.0   0.0
## betaht      1.9 0.7   0.5   1.4   1.9   2.4   3.4
## betasmoke   1.2 0.4   0.4   0.9   1.2   1.5   2.0
## betaui      0.9 0.5   0.0   0.6   0.9   1.2   1.9
## betarace2   1.4 0.5   0.3   1.0   1.4   1.7   2.5
## betarace3   0.9 0.4   0.1   0.6   0.9   1.2   1.8
## deviance  201.5 3.7 196.0 198.8 200.9 203.6 210.6
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 7.1 and DIC = 208.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
betaht <- mlgoutC$sims.list$betaht
acf(betaht)
```




```
y<-cumsum(betaht)/c(1:5000)
y<-data.frame(c=c(1:5000),y=y)
ggplot(y,aes(x=c, y=y))+geom_line(color = "black", size = 1)+ xlab("num. observations")+ylab("betaht value")
```



4.1 Comparation Bayesian models

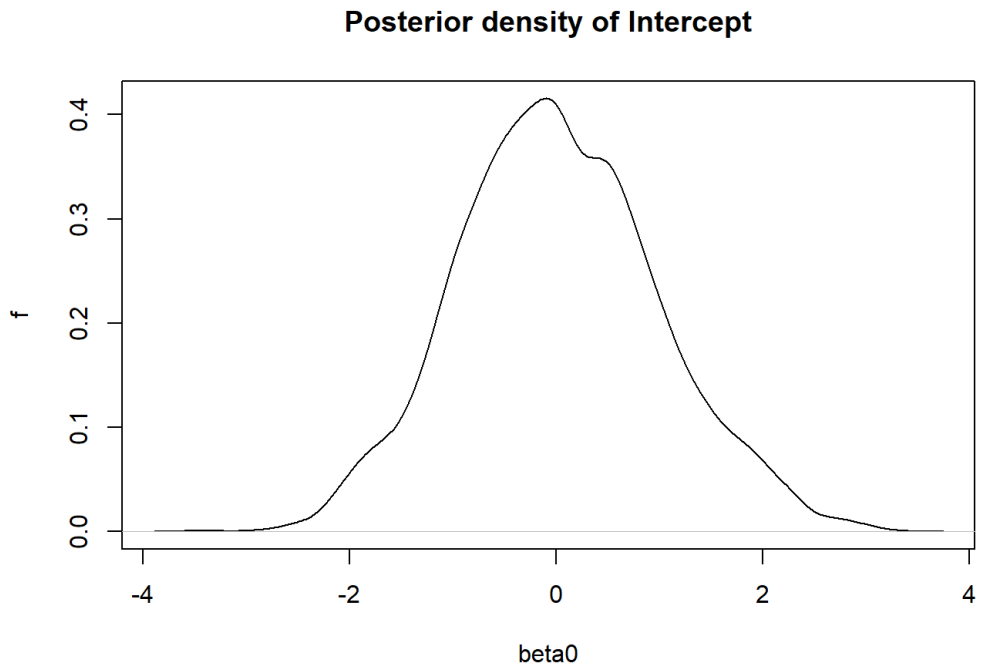
Now to compare the models, I use the DIC value. Similar to AIC or BIC for frequentist model, DIC could be used as measure to decide which model is better. DIC is the “Deviance Information Criterion” and is given by $DIC = D(\bar{\theta}) + 2p_D$ where $\bar{\theta}$ is the posterior mean of θ and p_D is equal to $\bar{D(\theta)} - D(\bar{\theta})$, where $\bar{D(\theta)}$ is the mean a posteriori of $D(\theta)$.
The best model results the one with the same variables of frequentist one. The value of DIC is 208.6, while the other two DIC values are 225.7 for the model with only **age** variable and 223.5for the second one with **age** and **lwt**.
Another possible way to compare models is using the Bayesian Factor. It is used to quantify the support for a model over another. Under the two hypothesis $H_0 : M_0$ and $H_1 : M_1$, where M_0 and M_1 are the two models estimated, the Bayesian factor to the model M_1 is $B_0^1 = \frac{P(M_1|data)}{P(M_0|data)} \frac{P(M_0)}{P(M_1)}$. The value represents confidence level to accept the null hypothesis: for values bigger than 1 we would prefer the model M_1 else for values smaller then 1 we prefer the other one. By the way, $P(M_1|data)$ for Bayes theorem is equal to $\frac{f(datos|M_1)P(M_1)}{f(datos)}$ so the

Bayesian factor is $B_0^1 = \frac{\frac{f(datos|M_1)P(M_1)}{f(datos)}}{\frac{f(datos|M_0)P(M_0)}{f(datos)}} \frac{P(M_0)}{P(M_1)} = \frac{f(datos|M_1)}{f(datos|M_0)}$. So, the Bayesian factor is equal to likelihood ratio. For this reason, we can see the likelihood value of each model. Mcmc routine gives us logmarglikelihood: for the first model the value is -125.247772, for the second model -132.8165891 and for the last one is -139.9808604. As you can see, if we calculate B_0^1 we will always obtain a value bigger then 1: also the Bayesian factor confirms **bM** model is the best of all models analyzed.

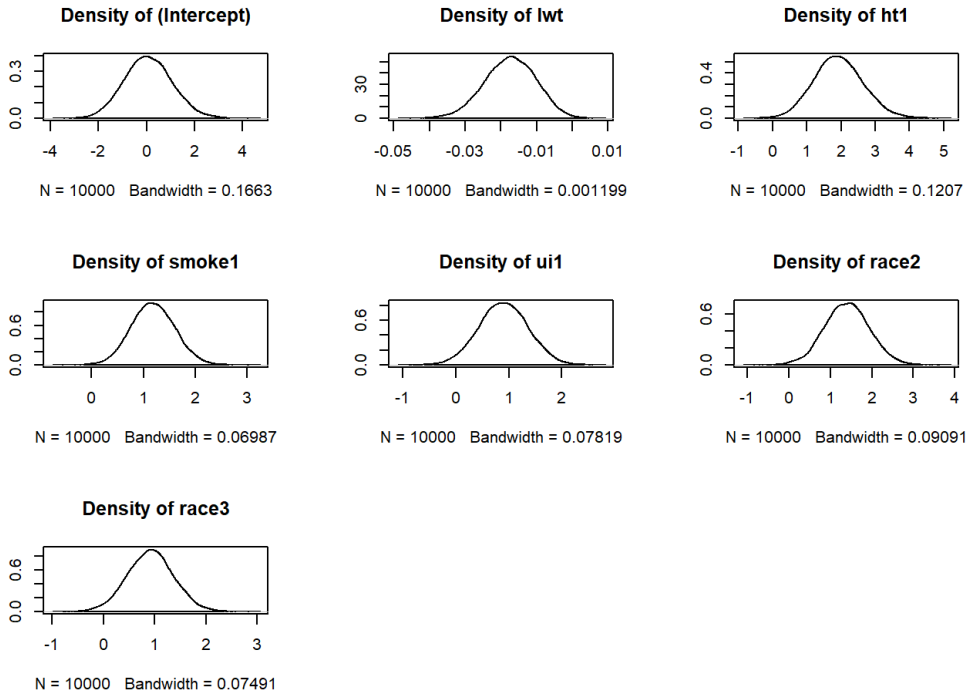
4.2 Posterior distribution of density

Using **plot(density())** or **densplot()** it is possible to show the posterior distribution of the parameters. Namely, the distribution of the values contained in Markov chain. For that, if I prefer use the command **densplot()** that will use data from **mcmclogit** method : there are more observation than OpenBugs model one. This could be visible comparing the first two plots of **intercept** density : one appears less smooth than the other one due to few observations.

```
plot(density(mlgoutC$sims.list$intercept),xlab=expression(beta0),ylab='f',main='Posterior density of Intercept')
```




```
par(mfrow=c(3,3))
densplot(bM)
```



4.3 Comparison between frequentist and Bayesian parameters

Now, we can compare coefficients between three methods (frequentist, MCMCpack and OpenBugs). Using the uninformative prior density for Bayesian method, we expect to find the same values: except for **intercept** other values are similar.

```
a<-fitfinal$coefficients
c<-summary(bM)[1]
b<-c$statistics[,1]
c<-mlgoutC$summary[1:7,1]
data.frame(FreqModel =a, MCMCpackModel=b, OpenBugsModel=c, row.names = c("intercept","lwt","ht","smoke","ui","race2","race
3"))
```

```
##             FreqModel MCMCpackModel OpenBugsModel
## intercept -0.03405073    0.07060079    0.01713488
## lwt       -0.01596480   -0.01743168   -0.01694420
## ht         1.81049984    1.92774619    1.91104372
## smoke      1.13145231    1.18481381    1.17862329
## ui         0.87713522    0.89582202    0.89050837
## race2      1.34380710    1.40101161    1.38461961
## race3      0.88163212    0.91800656    0.91432160
```

4.4 Some additional analysis

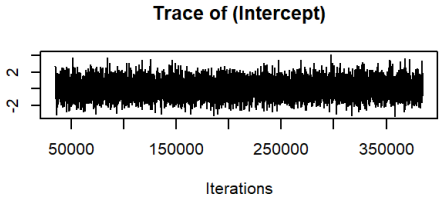
Now, using the **bM** model, I will look at what happens to the parameters when I change precision and the starting points of the chain.

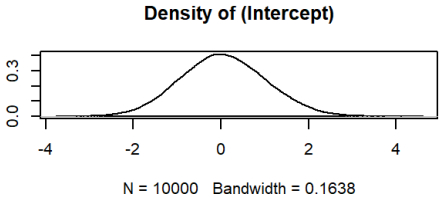
- Model with precision value equal to 0.01

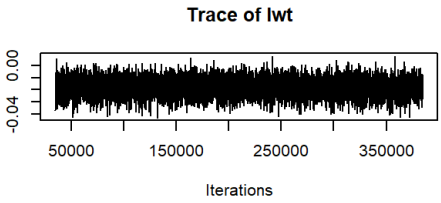
The ϕ is setted to 0.01, that means the variability is lower than before. Now the prior distribution could not be considered uninformative.

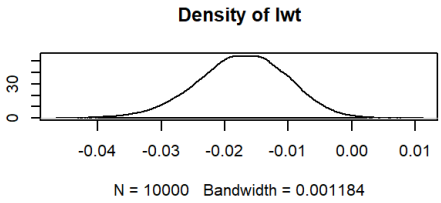
```
bM <- MCMClogit(low ~ lwt+ht+smoke+ui+race, b_train,burnin=1000*35,mcmc=10000*35, thin =35,
                        B0=0.01,marginal.likelihood="Laplace")
```

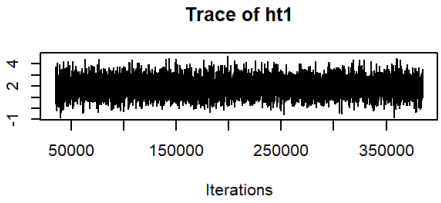
```
##
## Iterations = 35001:384966
## Thinning interval = 35
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  0.05534 0.975297 9.753e-03      1.032e-02
## lwt         -0.01731 0.007048 7.048e-05      7.426e-05
## ht1          1.91923 0.730571 7.306e-03      7.818e-03
## smoke1       1.18196 0.407625 4.076e-03      4.347e-03
## ui1          0.89604 0.463951 4.640e-03      4.840e-03
## race2        1.39637 0.548094 5.481e-03      5.899e-03
## race3        0.92149 0.445658 4.457e-03      4.706e-03
##
## 2. Quantiles for each variable:
##
##                  2.5%      25%      50%      75%    97.5%
## (Intercept) -1.789954 -0.60545  0.03360  0.70534  2.00021
## lwt         -0.031749 -0.02188 -0.01711 -0.01244 -0.00426
## ht1          0.532384  1.42353  1.90701  2.38494  3.39010
## smoke1       0.386512  0.90172  1.17964  1.45442  1.97663
## ui1         -0.002403  0.57918  0.89697  1.22048  1.78218
## race2        0.332656  1.02599  1.39133  1.76337  2.48545
## race3        0.070013  0.61927  0.91163  1.21553  1.81074
```

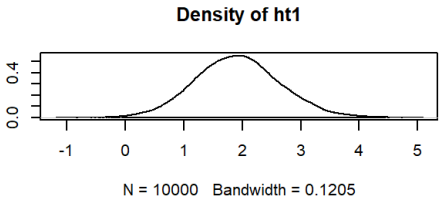

Trace of (Intercept)
-2
2
50000
150000
250000
350000
Iterations

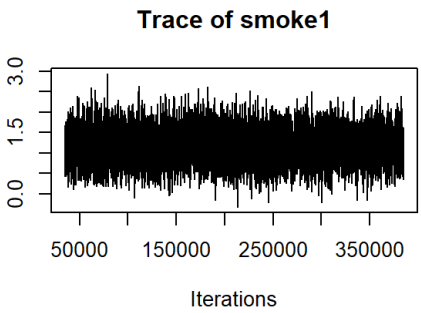

Density of (Intercept)
0.0
0.3
-4
-2
0
2
4
N = 10000 Bandwidth = 0.1638

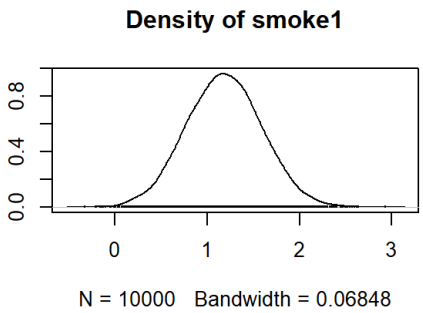

Trace of lwt
-0.04
0.00
50000
150000
250000
350000
Iterations

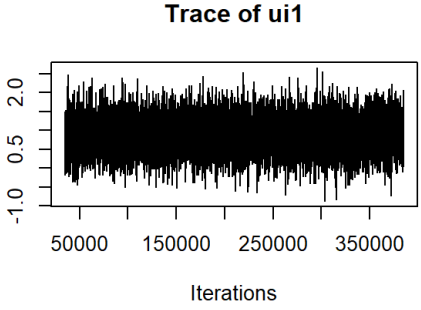

Density of lwt
0
30
-0.04
-0.03
-0.02
-0.01
0.00
0.01
N = 10000 Bandwidth = 0.001184

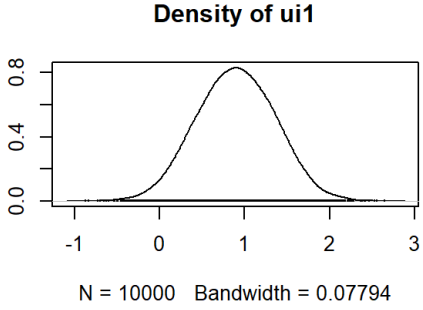

Trace of ht1
-1
2
4
50000
150000
250000
350000
Iterations

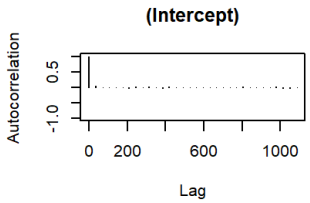

Density of ht1
0.0
0.4
-1
0
1
2
3
4
5
N = 10000 Bandwidth = 0.1205

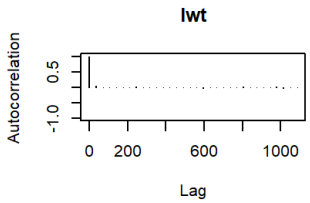

Trace of smoke1
0.0
1.5
3.0
50000
150000
250000
350000
Iterations

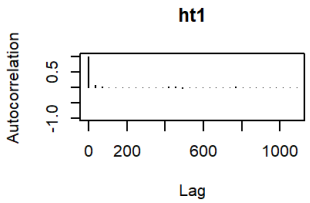

Density of smoke1
0.0
0.4
0.8
0
1
2
3
N = 10000 Bandwidth = 0.06848

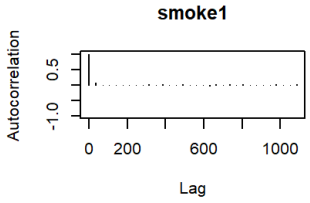

Trace of ui1
-1.0
0.5
2.0
50000
150000
250000
350000
Iterations

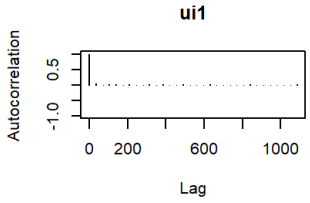

Density of ui1
0.0
0.4
0.8
-1
0
1
2
3
N = 10000 Bandwidth = 0.07794

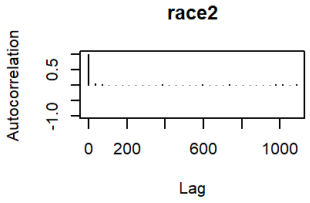

(Intercept)
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

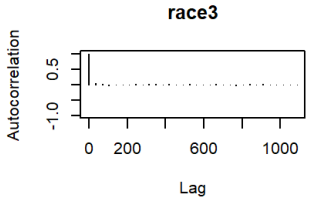

lwt
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

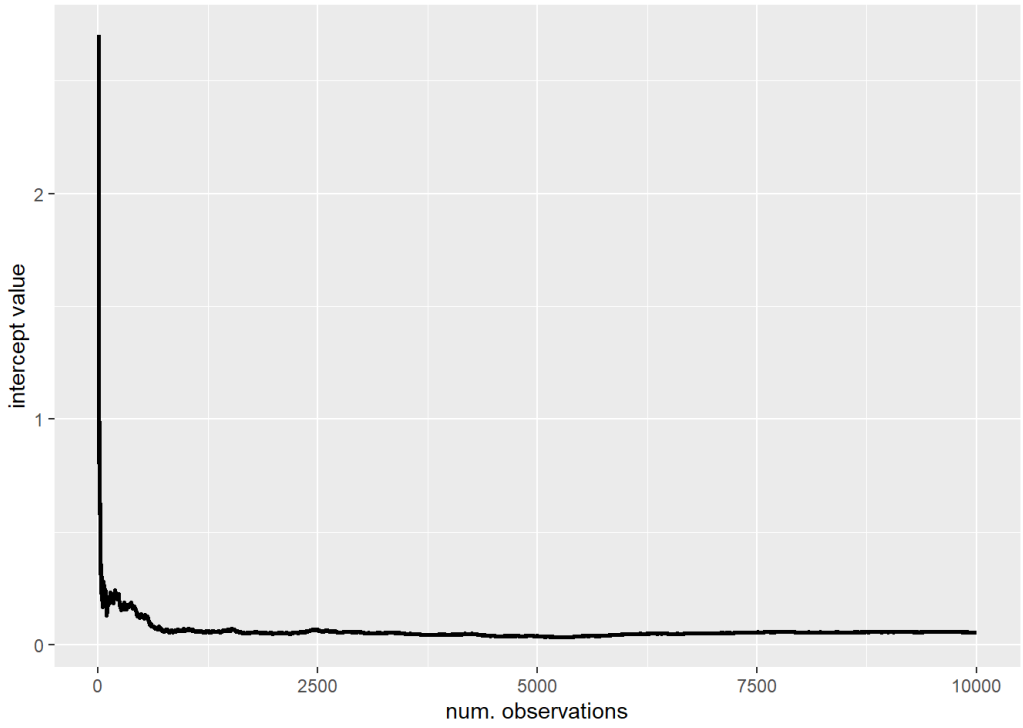

ht1
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

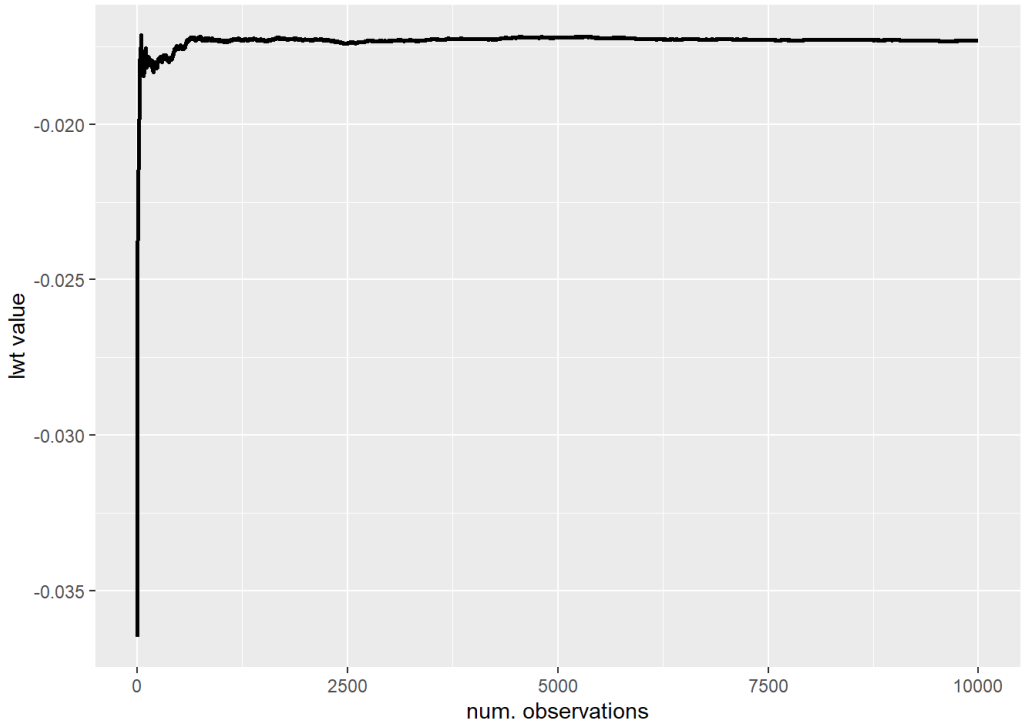

smoke1
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

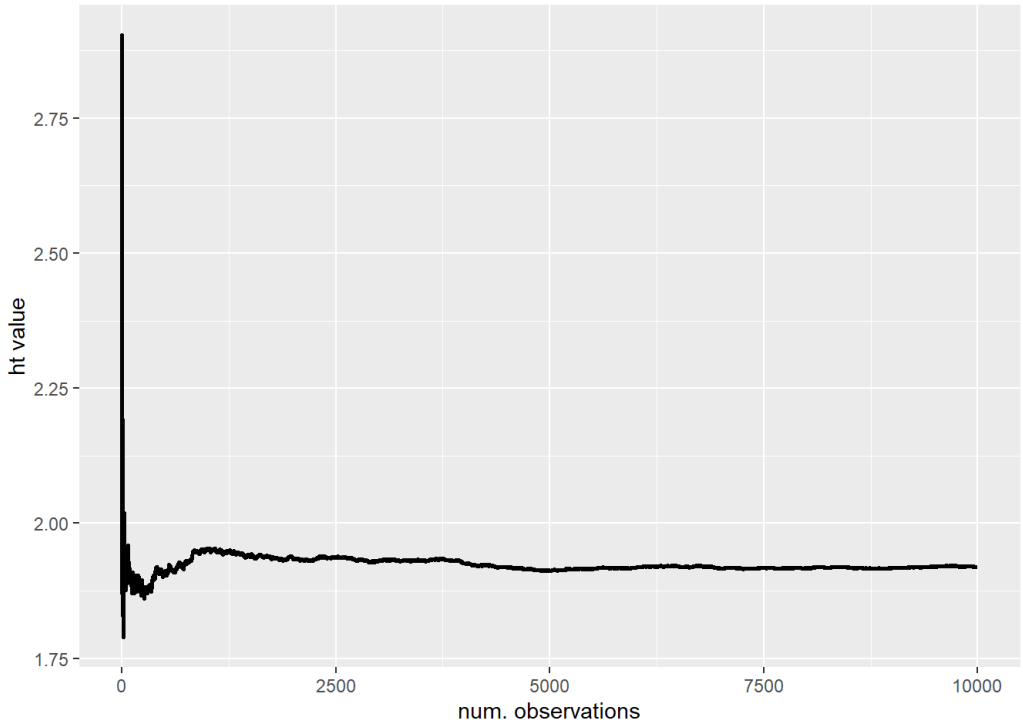

ui1
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

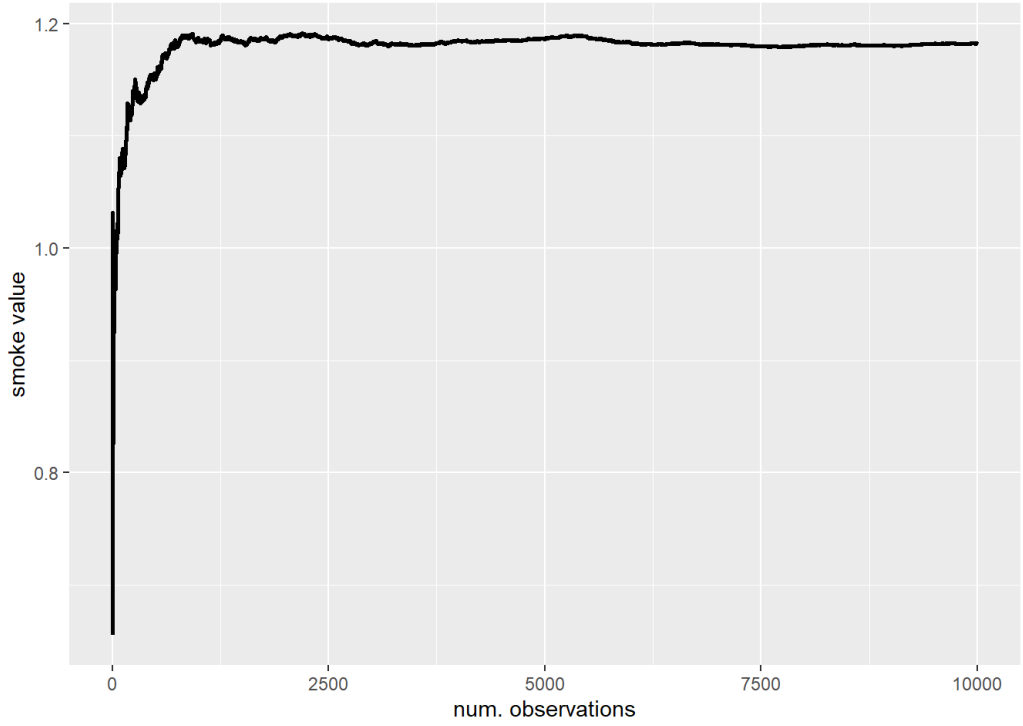

race2
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

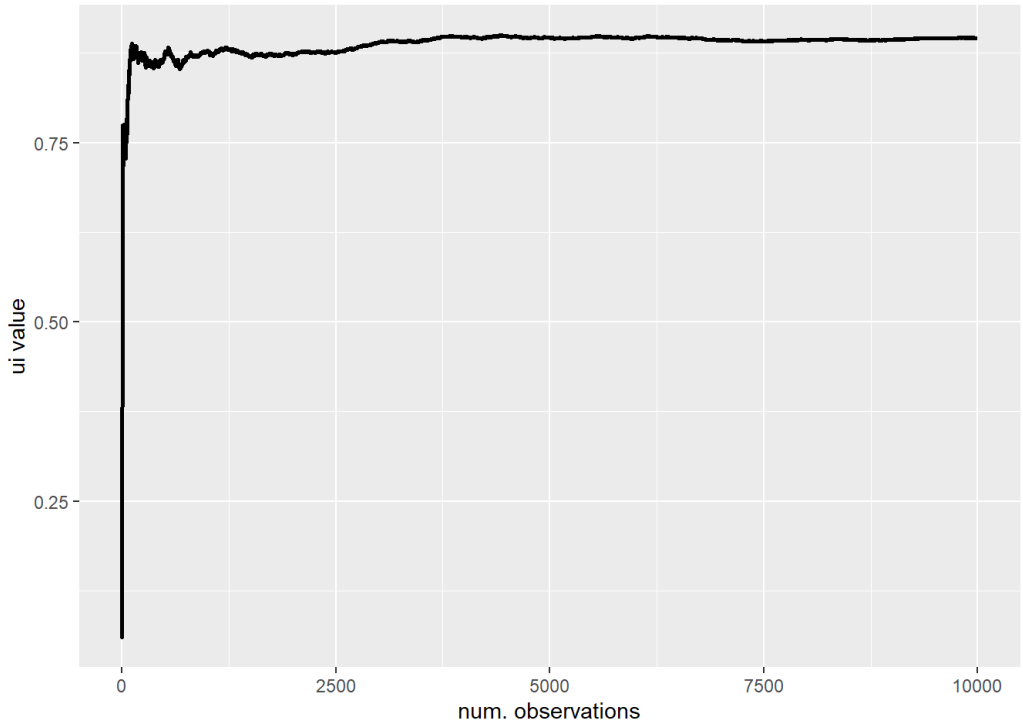

race3
Autocorrelation
1.0
0.5
0
200
600
1000
Lag

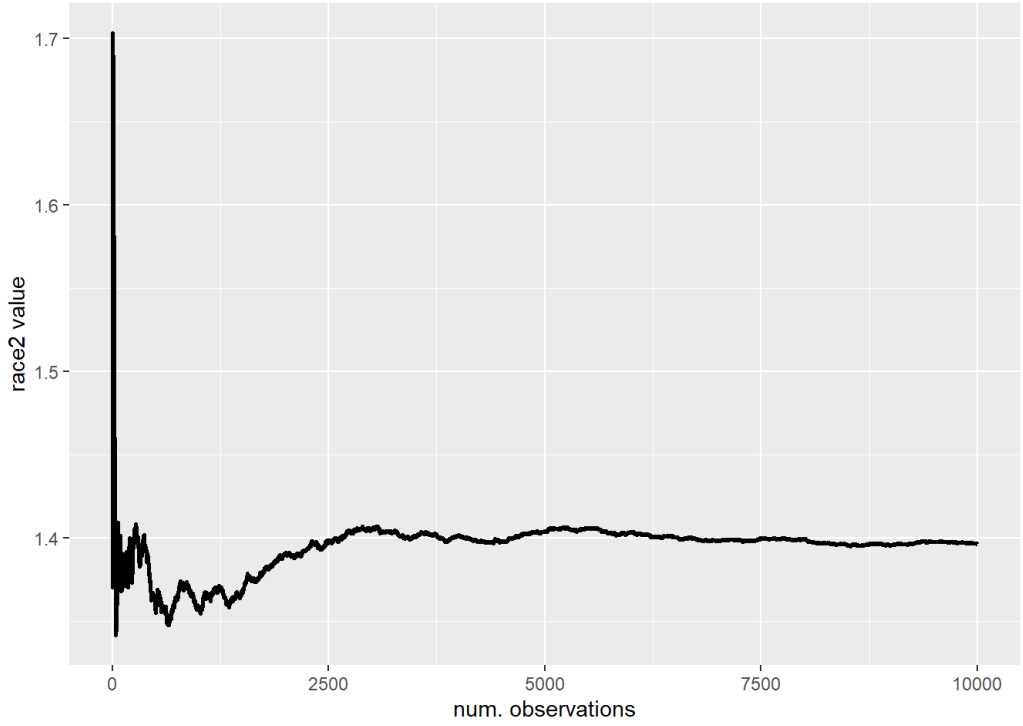

intercept value
2
1
0
0
2500
5000
7500
10000
num. observations

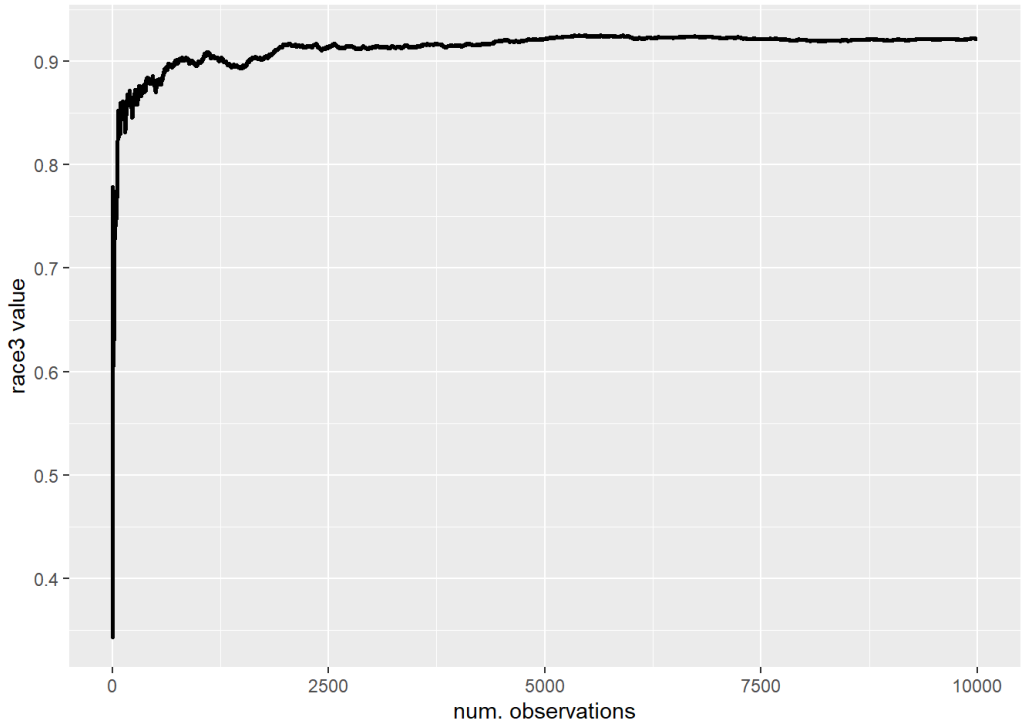

-0.020
-0.025
-0.030
-0.035
lwt value
0
2500
5000
7500
10000
num. observations


2.75
2.50
2.25
2.00
1.75
ht value
0
2500
5000
7500
10000
num. observations


1.2
1.0
0.8
smoke value
0
2500
5000
7500
10000
num. observations


0.75
0.50
0.25
ui value
0
2500
5000
7500
10000
num. observations



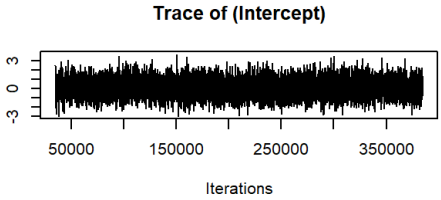


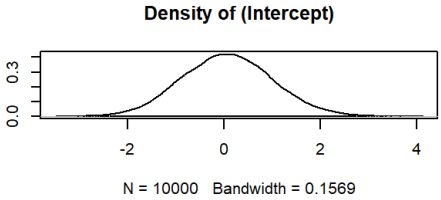
- Model with precision value equal to 0.1

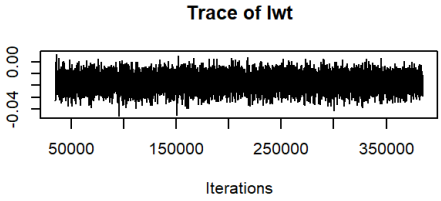
The ϕ is setted to 0.1, that means σ^2 is equal to 10.

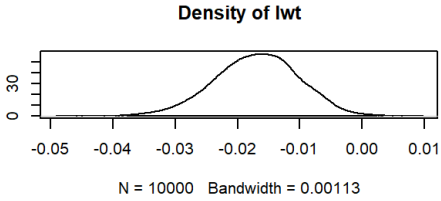
```
bM <- MCMClogit(low ~ lwt+ht+smoke+ui+race, b_train,burnin=1000*35,mcmc=10000*35, thin =35,
                        B0=0.1,marginal.likelihood="Laplace")
```

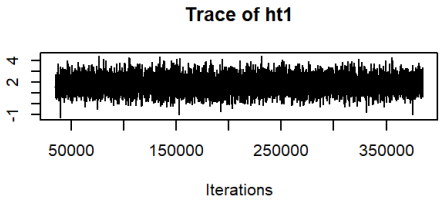
```
##
## Iterations = 35001:384966
## Thinning interval = 35
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  0.06885 0.934041 9.340e-03      9.945e-03
## lwt         -0.01696 0.006726 6.726e-05      7.239e-05
## ht1          1.82154 0.699173 6.992e-03      7.477e-03
## smoke1       1.14243 0.399427 3.994e-03      4.226e-03
## ui1          0.86856 0.458326 4.583e-03      4.804e-03
## race2        1.33955 0.530335 5.303e-03      5.502e-03
## race3        0.88384 0.438005 4.380e-03      4.648e-03
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%     97.5%
## (Intercept) -1.72307 -0.57389  0.05943  0.69374  1.940764
## lwt         -0.03073 -0.02139 -0.01680 -0.01237 -0.004489
## ht1          0.48190  1.34515  1.81567  2.27971  3.225457
## smoke1       0.38408  0.86599  1.13526  1.41403  1.931528
## ui1         -0.02735  0.56102  0.86611  1.17446  1.777492
## race2        0.30746  0.98744  1.34317  1.68878  2.388326
## race3        0.03611  0.58655  0.88473  1.17280  1.758983
```

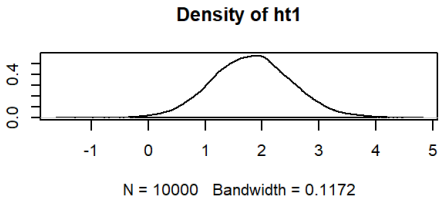

Trace of (Intercept)
Iterations

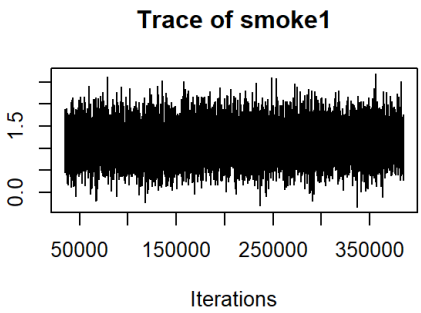

Density of (Intercept)
N = 10000 Bandwidth = 0.1569

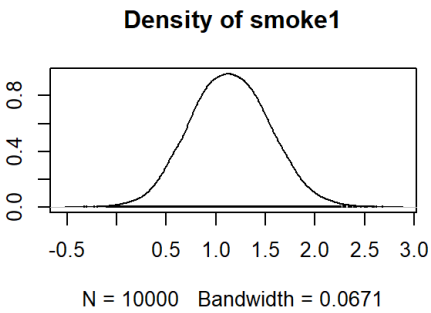

Trace of lwt
Iterations

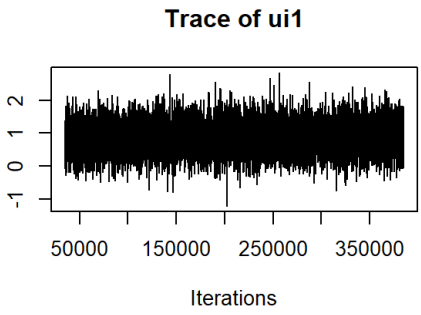

Density of lwt
N = 10000 Bandwidth = 0.00113

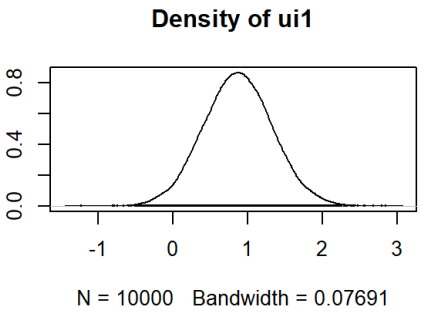

Trace of ht1
Iterations

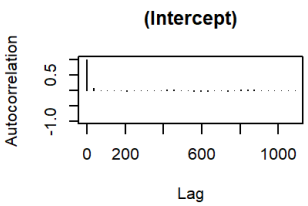

Density of ht1
N = 10000 Bandwidth = 0.1172

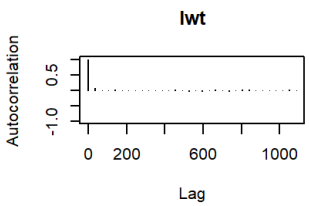

Trace of smoke1
Iterations

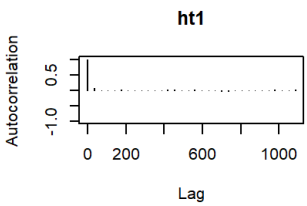

Density of smoke1
N = 10000 Bandwidth = 0.0671

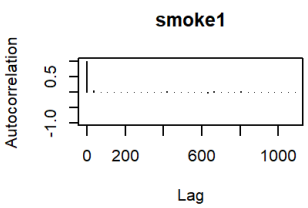

Trace of ui1
Iterations

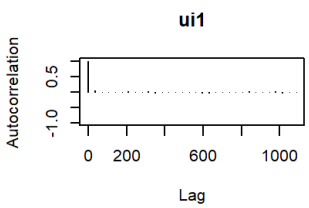

Density of ui1
N = 10000 Bandwidth = 0.07691

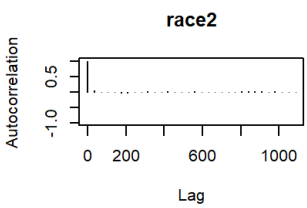

(Intercept)
Autocorrelation
Lag

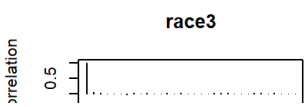

lwt
Autocorrelation
Lag

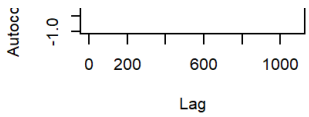

ht1
Autocorrelation
Lag

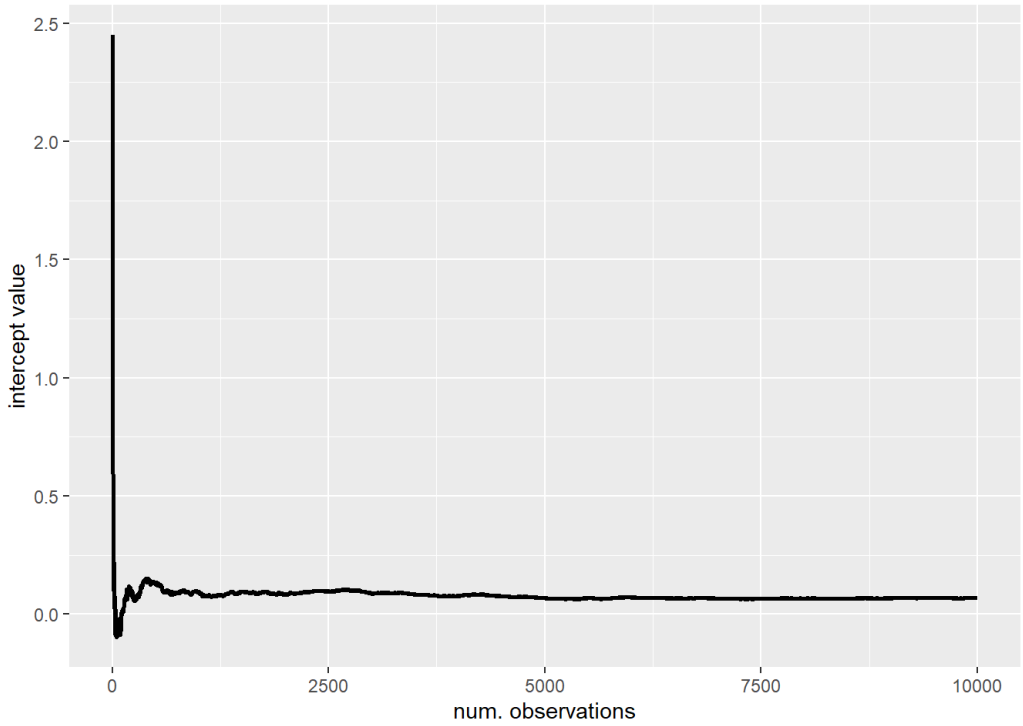

smoke1
Autocorrelation
Lag

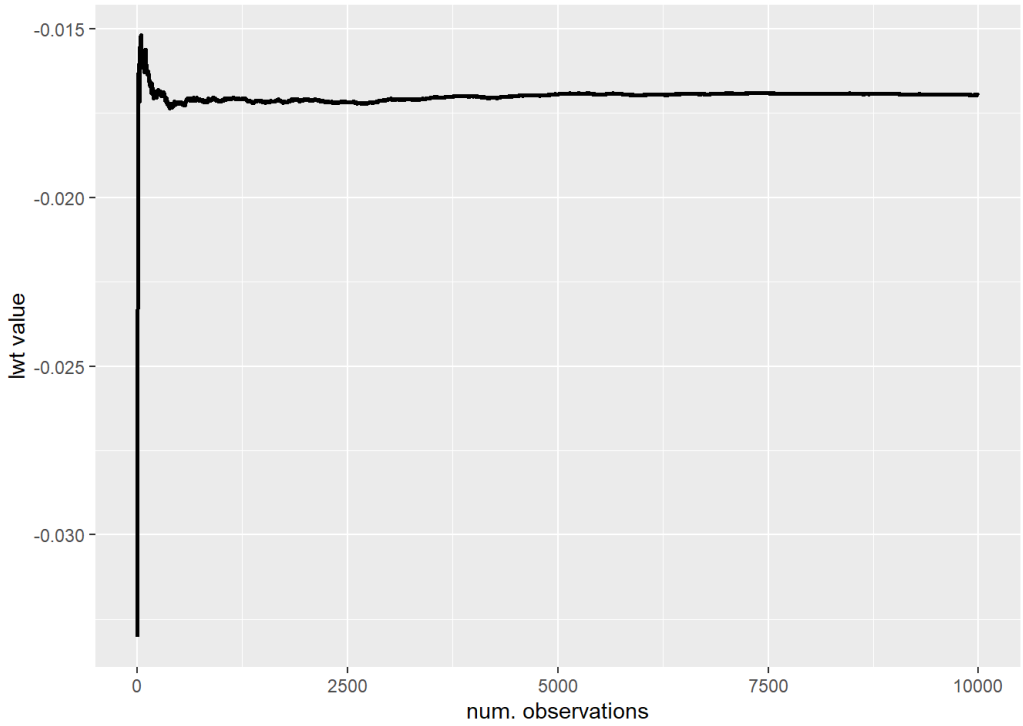

ui1
Autocorrelation
Lag

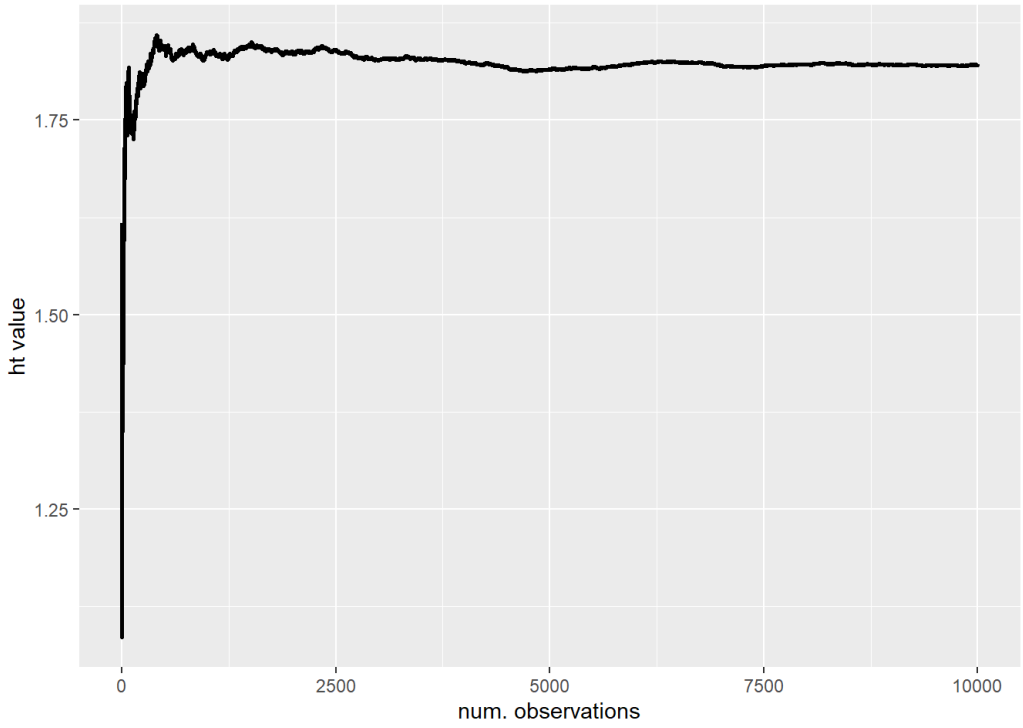

race2
Autocorrelation
Lag

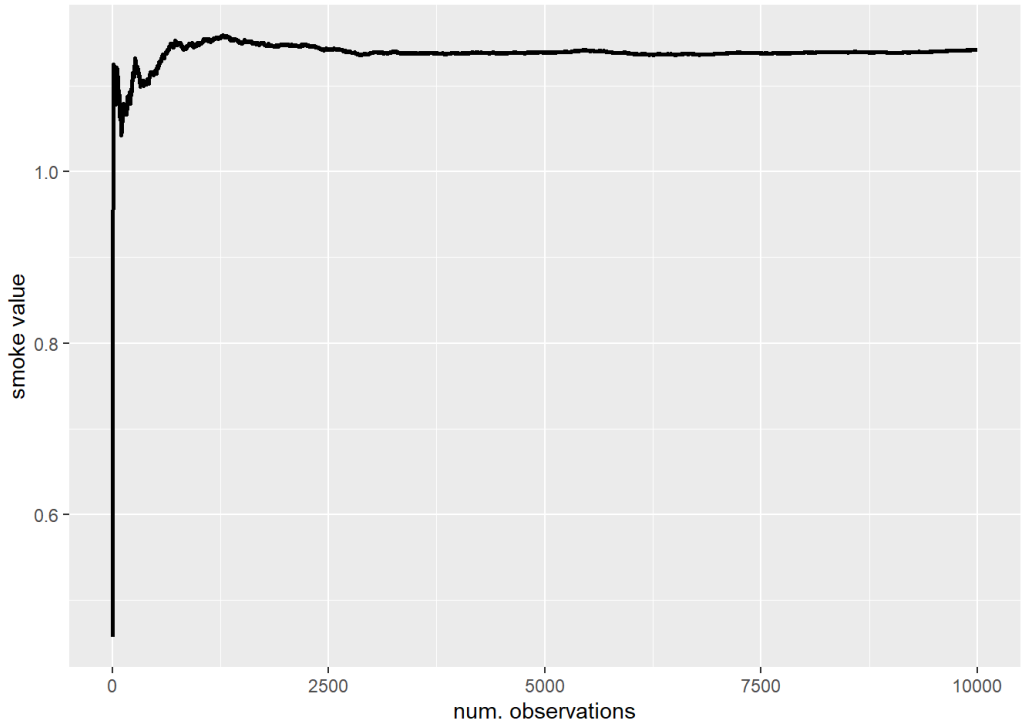

race3

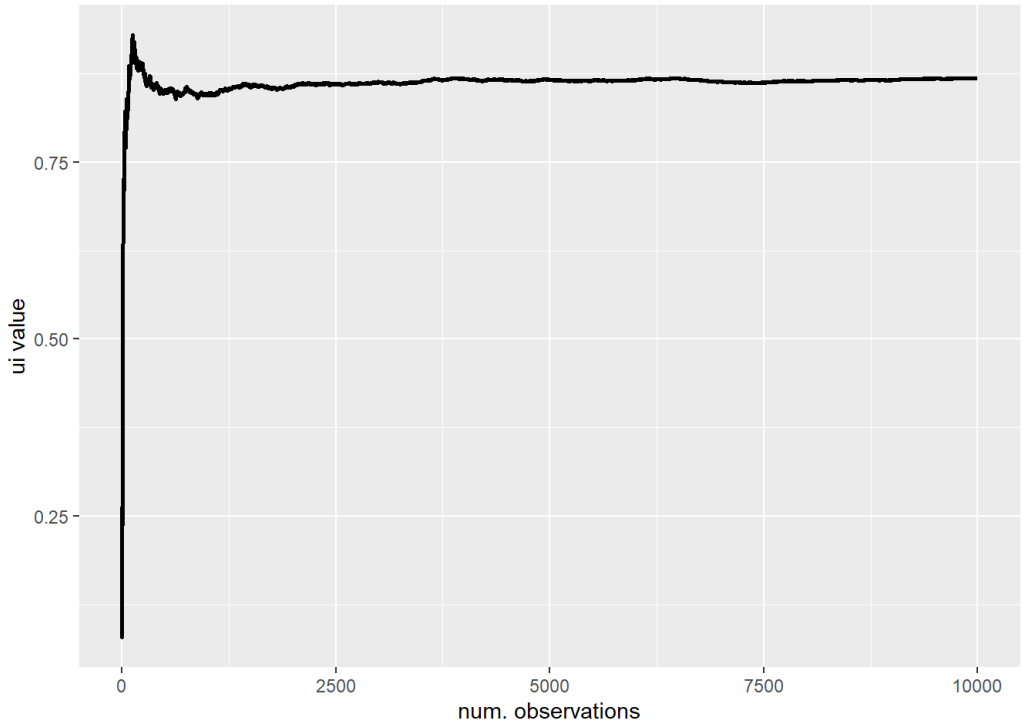

Autocc
-1.0
0
200
600
1000
Lag

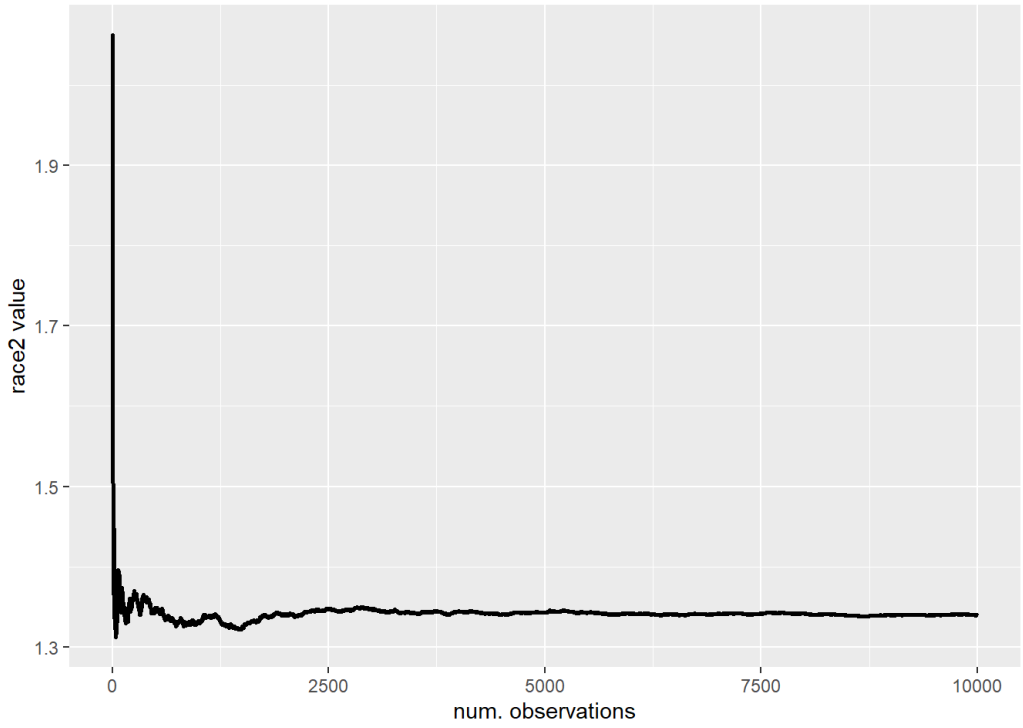

2.5
2.0
1.5
1.0
0.5
0.0
intercept value
0
2500
5000
7500
10000
num. observations

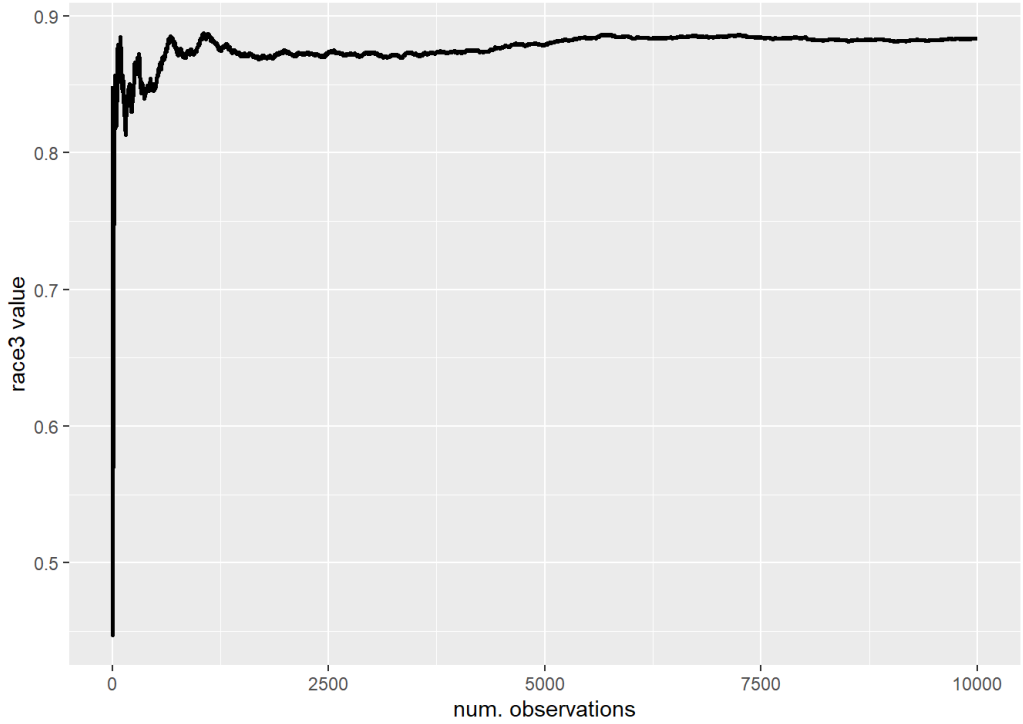

-0.015
-0.020
-0.025
-0.030
lwt value
0
2500
5000
7500
10000
num. observations


1.75
1.50
1.25
ht value
0
2500
5000
7500
10000
num. observations


1.0
0.8
0.6
smoke value
0
2500
5000
7500
10000
num. observations


ui value
0.75
0.50
0.25
0
2500
5000
7500
10000
num. observations


race2 value
1.9
1.7
1.5
1.3
0
2500
5000
7500
10000
num. observations

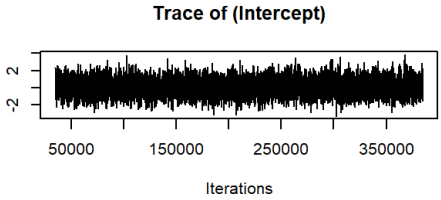


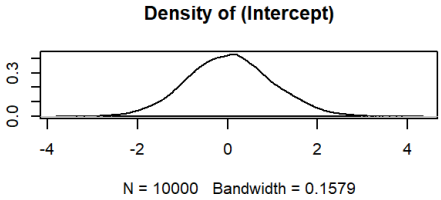
- Model with different starting chain values and precision value equal to 0.1

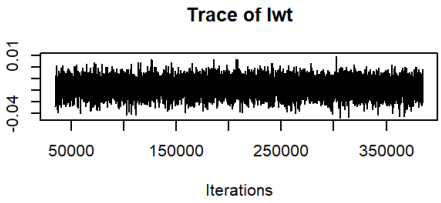
The β are setted to values different from frequentist parameters values, even if we expect the algorithm should not converge to different values: the prior distributions are not uninformative. But the corvengence should be faster.

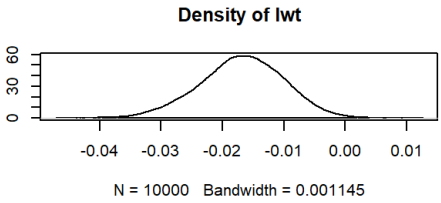
```
beta.start <- c(-0.03,-0.01,1.8,1.1,0.9,1.3,0.9)
bM <- MCMClogit(low ~ lwt+ht+smoke+ui+race, b_train,burnin=1000*35,mcmc=10000*35, thin =35,
                        B0=0.1,marginal.likelihood="Laplace", beta.start = beta.start )
```

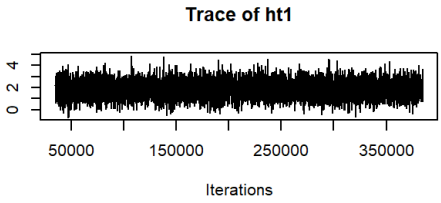
```
##
## Iterations = 35001:384966
## Thinning interval = 35
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## (Intercept)  0.05793 0.942724 9.427e-03      9.840e-03
## lwt         -0.01690 0.006826 6.826e-05      7.165e-05
## ht1          1.82186 0.707061 7.071e-03      7.404e-03
## smoke1       1.14508 0.400933 4.009e-03      4.316e-03
## ui1          0.87923 0.452375 4.524e-03      4.752e-03
## race2        1.33579 0.529739 5.297e-03      5.570e-03
## race3        0.87955 0.440408 4.404e-03      4.658e-03
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%     75%     97.5%
## (Intercept) -1.75421 -0.58433  0.04852  0.6751  1.941808
## lwt         -0.03103 -0.02133 -0.01670 -0.0122 -0.004246
## ht1          0.47340  1.34205  1.80065  2.2769  3.258898
## smoke1       0.36053  0.86895  1.14522  1.4160  1.932240
## ui1         -0.02950  0.58037  0.88105  1.1810  1.747754
## race2        0.30923  0.98156  1.33134  1.6861  2.373934
## race3        0.02400  0.58274  0.87592  1.1678  1.751800
```

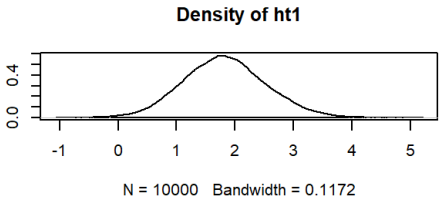

Trace of (Intercept)
Iterations

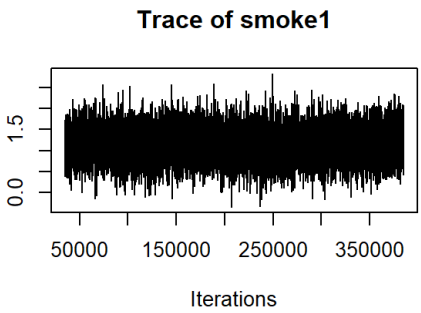

Density of (Intercept)
N = 10000 Bandwidth = 0.1579

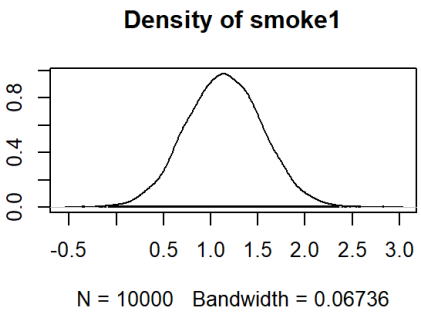

Trace of lwt
Iterations

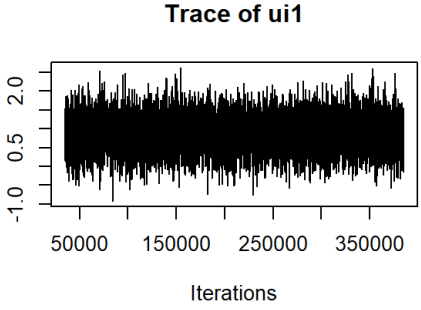

Density of lwt
N = 10000 Bandwidth = 0.001145

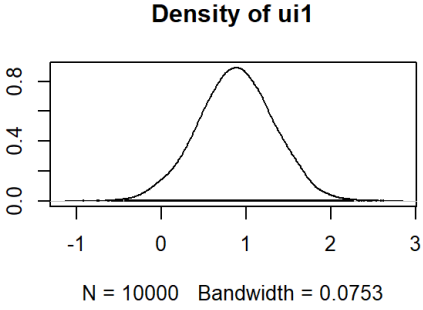

Trace of ht1
Iterations

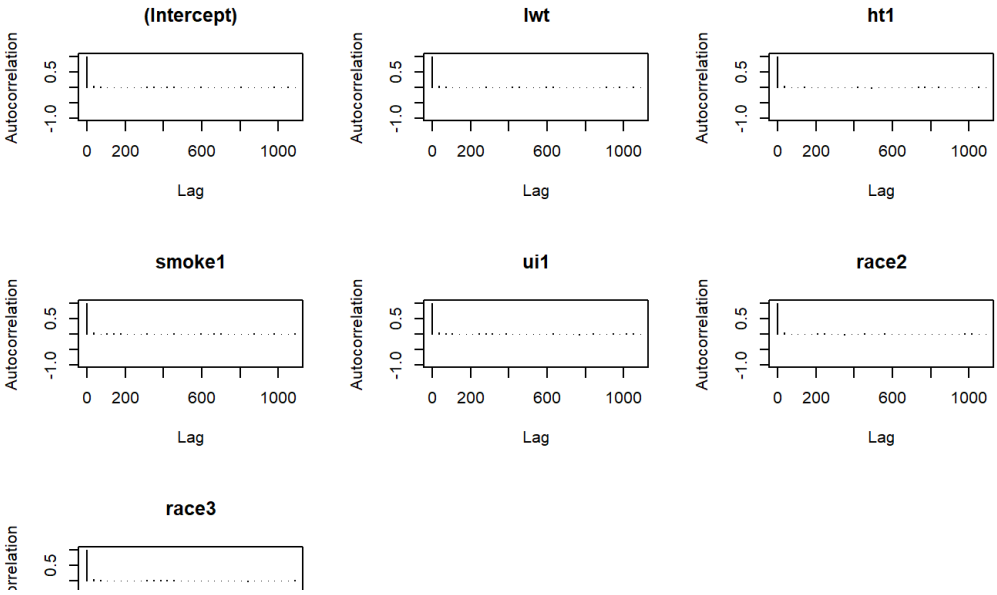

Density of ht1
N = 10000 Bandwidth = 0.1172

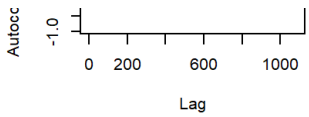

Trace of smoke1
Iterations

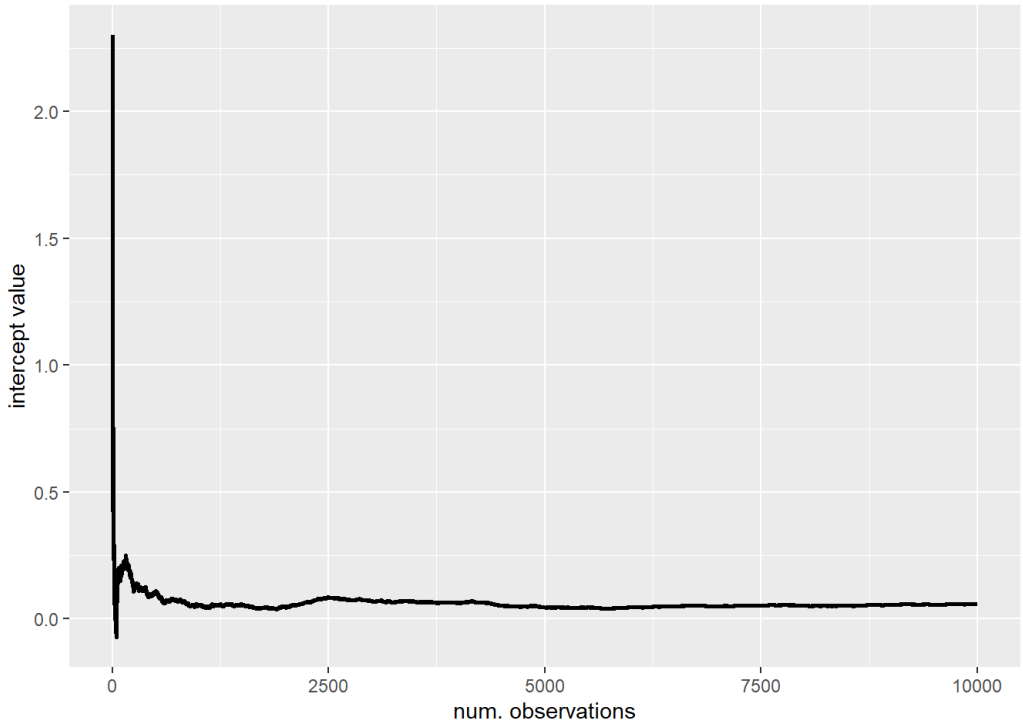

Density of smoke1
N = 10000 Bandwidth = 0.06736

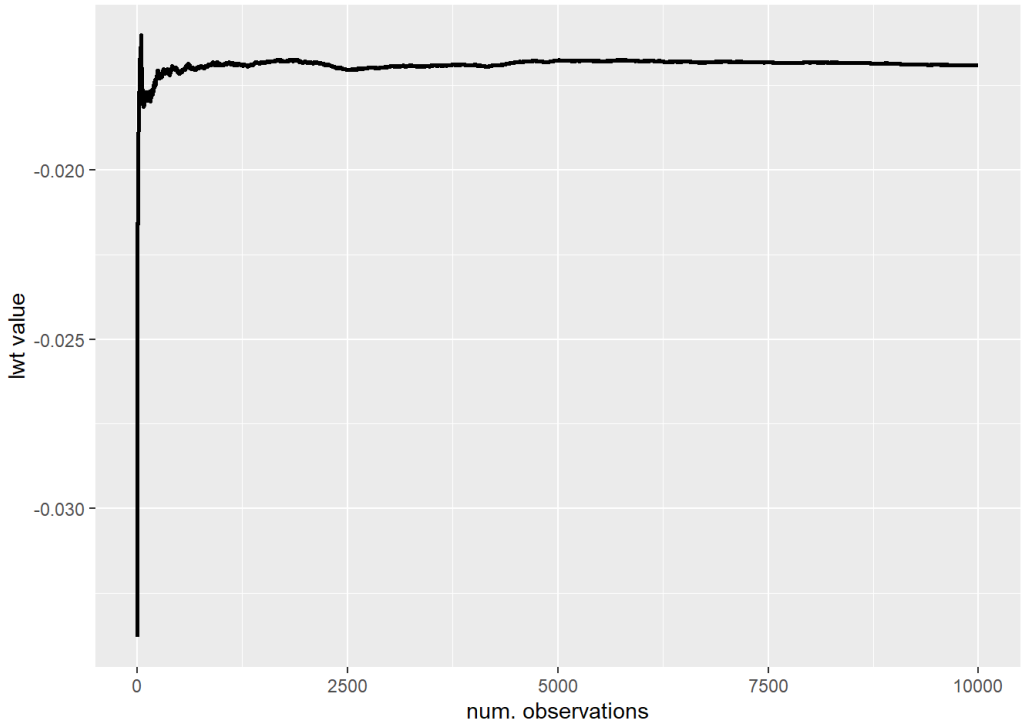

Trace of ui1
Iterations

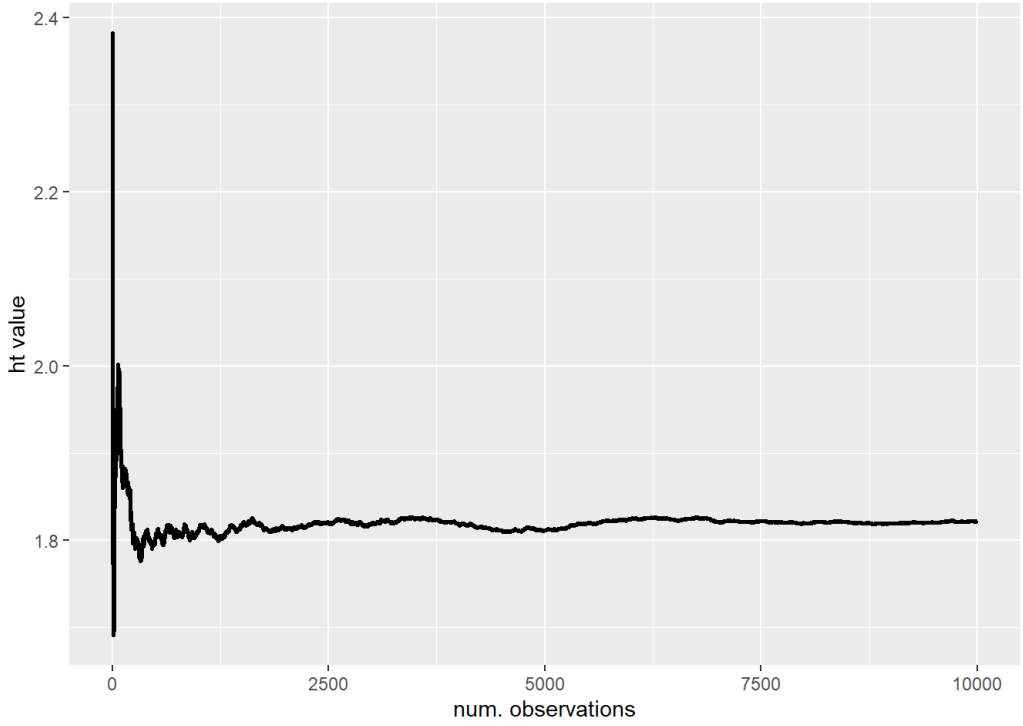

Density of ui1
N = 10000 Bandwidth = 0.0753

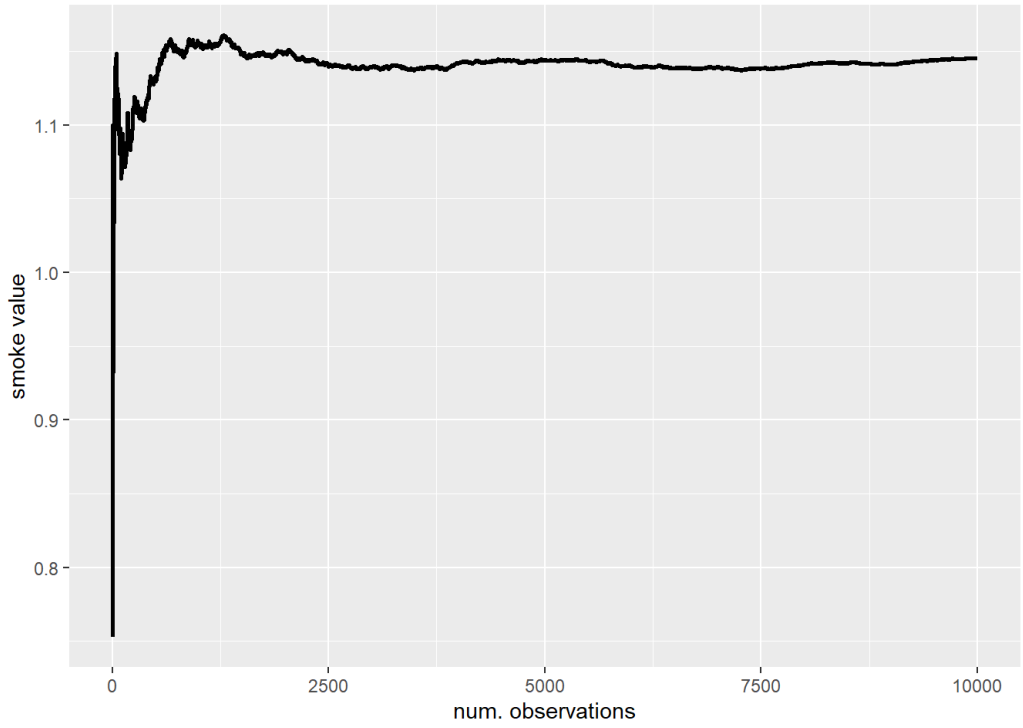

(Intercept)
lwt
ht1
smoke1
ui1
race2
race3
Autocorrelation
Lag

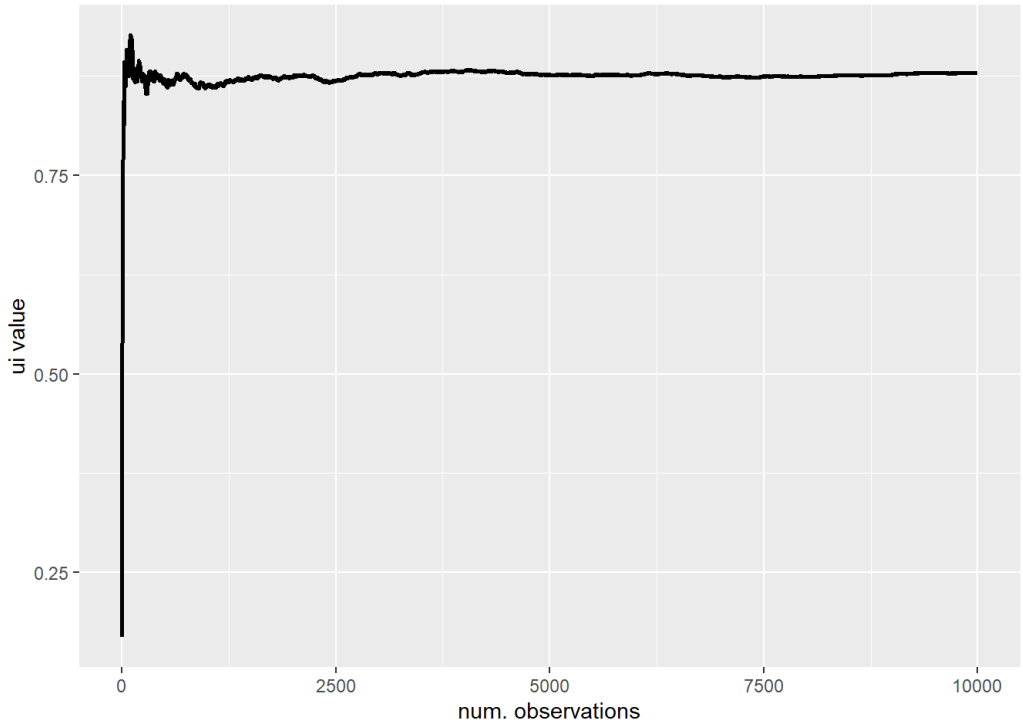

Autocc
-1.0
0
200
600
1000
Lag

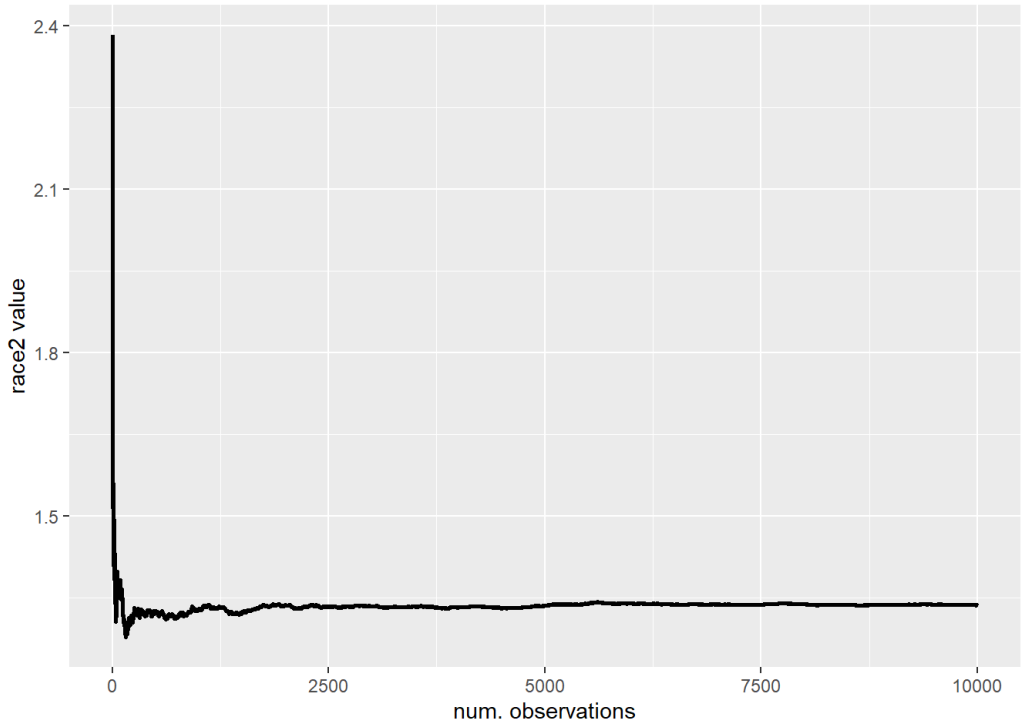

2.0
1.5
1.0
0.5
0.0
intercept value
0
2500
5000
7500
10000
num. observations

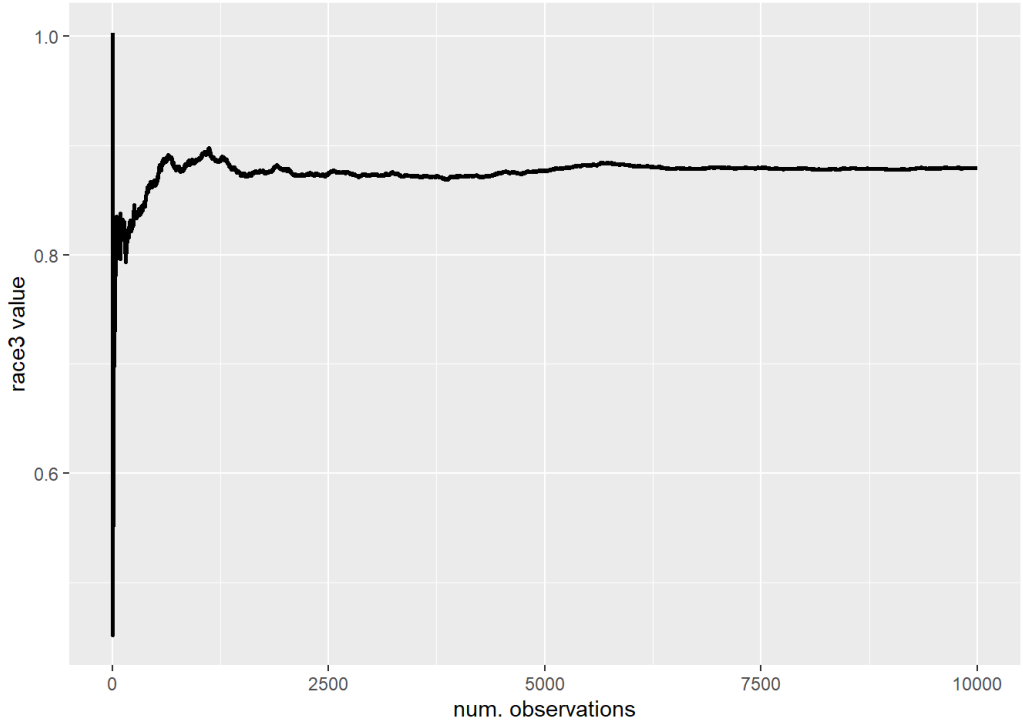

-0.020
-0.025
-0.030
lwt value
0
2500
5000
7500
10000
num. observations


2.4
2.2
2.0
1.8
ht value
0
2500
5000
7500
10000
num. observations


1.1
1.0
0.9
0.8
smoke value
0
2500
5000
7500
10000
num. observations


ui value
0.75
0.50
0.25
0
2500
5000
7500
10000
num. observations


race2 value
2.4
2.1
1.8
1.5
0
2500
5000
7500
10000
num. observations



4.5 Prediction with last model

To make prediction, I need to set a function which model parameters. Then using **b_test** values I will obtain probability for each observation to be evaluated as 0 or 1.

```
par(mfrow=c(3,2))
c<-summary(bM)[1]
int <- c$statistics[1,1]
lwt <-c$statistics[2,1]
ht1 <- c$statistics[3,1]
smoke1<-c$statistics[4,1]
ui1<-c$statistics[5,1]
race2<-c$statistics[6,1]
race3 <-c$statistics[7,1]

ht <- as.numeric(b_test$ht)-1
smoke <- as.numeric(b_test$smoke)-1
ui <- as.numeric(b_test$ui)-1
race <- as.numeric(b_test$race)-1
race21 <- race
race21[race21!=1] <- 0
race31 <- race
race31[race31!=2] <- 0
race31[race31==2] <- 1
test <- data.frame(low = b_test$low,lwt=b_test$lwt, ht=ht,smoke=smoke, ui=ui, race2=race21,race3=race31 )
plow <- rep(NA,9)
mupred <- rep(NA,9)

  for( i in 1 : length(test$low) ) {
    mupred[i] <- (int+lwt*test[i,2]+ht1*test[i,3]+smoke1*test[i,4]+ui1*test[i,5]+race2*test[i,6]+race3*test[i,7])[1]
    plow[i] <- exp(mupred[i])/(1+exp(mupred[i]))
    }
```

The probabilities are

```
plow
```

```
## [1] 0.3738280 0.4211953 0.2906101 0.3888285 0.2538263 0.2515500 0.2210859
## [8] 0.2677890 0.3620438
```

4.6 Different link function for frequentist and Bayesian methods

It is possible to change the link function when the model is created. Different link function means different way to compute **p_{i}** values.

- Probit

The probit funcion is $\Phi(p_i)^{-1} = x_i^T\beta$. That means $p_i = \Phi(x_i^T\beta)$, where $\Phi()$ is the standard normal distribution.

```
fitprobit <- glm(low ~ lwt+ht+smoke+ui+race, family=binomial(link = "probit"),
            data=b_train)
summary(fitprobit)
```

```
##
## Call:
## glm(formula = low ~ lwt + ht + smoke + ui + race, family = binomial(link = "probit"),
##     data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6678  -0.8539  -0.5174   0.9732   2.2011
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.035051   0.556740  -0.063  0.94980
## lwt         -0.009520   0.003952  -2.409  0.01600 *
## ht1          1.092200   0.412913   2.645  0.00817 **
## smoke1       0.690374   0.233744   2.954  0.00314 **
## ui1          0.531915   0.274563   1.937  0.05271 .
## race2        0.804594   0.315063   2.554  0.01066 *
## race3        0.525298   0.256977   2.044  0.04094 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 193.94  on 173  degrees of freedom
## AIC: 207.94
##
## Number of Fisher Scoring iterations: 5
```

```
bprobit <- MCMCprobit(low ~ lwt+ht+smoke+ui+race, b_train,burnin=1000*35,mcmc=10000*35, thin =35,
                      B0=0.0001,marginal.likelihood="Laplace")



summary(bprobit)
```

```
##
## Iterations = 35001:384966
## Thinning interval = 35
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                  Mean       SD  Naive SE Time-series SE
## (Intercept) -0.011696 0.559149 5.591e-03      5.493e-03
## lwt         -0.009892 0.004013 4.013e-05      3.939e-05
## ht1          1.116556 0.425581 4.256e-03      4.256e-03
## smoke1       0.703643 0.236668 2.367e-03      2.323e-03
## ui1          0.537946 0.270446 2.704e-03      2.704e-03
## race2        0.825689 0.321136 3.211e-03      3.189e-03
## race3        0.538627 0.259190 2.592e-03      2.592e-03
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%       50%       75%     97.5%
## (Intercept) -1.11087 -0.38988 -0.015884  0.377039  1.073636
## lwt         -0.01781 -0.01263 -0.009784 -0.007167 -0.002179
## ht1          0.28725  0.82233  1.111018  1.402486  1.964571
## smoke1       0.24607  0.54557  0.702994  0.864457  1.171621
## ui1          0.01483  0.35303  0.537065  0.716683  1.073633
## race2        0.19730  0.60939  0.825605  1.040631  1.451867
## race3        0.03247  0.36266  0.541528  0.711978  1.048617
```

- Cloglog

The complementary log-log (cloglog) function is $log(-log(1-p_i)) = x_i^T\beta$. Namely, p_i is equal to $1 - e^{-e^{x_i^T\beta}}$. I will use **bugs()** function to create Bayesian model.

```
fitcloglog <- glm(low ~ lwt+ht+smoke+ui+race, family=binomial(link = "cloglog"),
            data=b_train)
summary(fitcloglog)
```

```
##
## Call:
## glm(formula = low ~ lwt + ht + smoke + ui + race, family = binomial(link = "cloglog"),
##     data = b_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7858  -0.8214  -0.5482   0.9618   2.0997
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.497502   0.749763  -0.664  0.50698
## lwt         -0.011949   0.005353  -2.232  0.02560 *
## ht1          1.412928   0.454341   3.110  0.00187 **
## smoke1       0.862547   0.303406   2.843  0.00447 **
## ui1          0.664034   0.332957   1.994  0.04611 *
## race2        1.103245   0.390382   2.826  0.00471 **
## race3        0.686359   0.340765   2.014  0.04399 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.76  on 179  degrees of freedom
## Residual deviance: 194.79  on 173  degrees of freedom
## AIC: 208.79
##
## Number of Fisher Scoring iterations: 6
```

```
y <- b_train$low
n <- length(y)
lwt<-b_train$lwt
ht <- as.numeric(b_train$ht)-1
smoke <- as.numeric(b_train$smoke)-1
ui <- as.numeric(b_train$ui)-1
race <- as.numeric(b_train$race)-1
race2 <- race
race2[race2!=1] <- 0
race3 <- race
race3[race3!=2] <- 0
race3[race3==2] <-1

mlg <- function(){
  for (i in 1:n) {
    y[i] ~ dbern(p[i])
    cloglog(p[i]) <- intercepto+betalwt*lwt[i]+betaht*ht[i]+betasmoke*smoke[i]+betaui*ui[i]+betarace2*race2[i]+betarace3*ra
ce3[i] }
  intercepto ~ dnorm(0,0.0001)
  betalwt ~ dnorm(0,0.0001)
  betaht ~ dnorm(0,0.0001)
  betasmoke ~ dnorm(0,0.0001)
  betaui ~ dnorm(0,0.0001)
  betarace2 ~ dnorm(0,0.0001)
  betarace3 ~ dnorm(0,0.0001)
}

mlgdata <- list(n = n,y = y,lwt=lwt,ht=ht,smoke=smoke,ui=ui,race2=race2,race3=race3)

mlginits <- function() {
  list(intercepto=0,betalwt=0,betaht=1.9,betasmoke=1.1,betaui=0.9,betarace2=1.4,betarace3=0.9)
}
mlgoutCloglog <- bugs(data=mlgdata,inits=mlginits,parameters.to.save=c("intercepto","betalwt","betaht","betasmoke","betaui"
,"betarace2","betarace3"),model.file=mlg,
               n.chains = 1,n.iter = 10000)
mlgoutCloglog
```

```
## Inference for Bugs model at "C:/Users/franc/AppData/Local/Temp/RtmpWafK5Z/model395418a3348.txt",
## Current: 1 chains, each with 10000 iterations (first 5000 discarded)
## Cumulative: n.sims = 5000 iterations saved
##             mean  sd  2.5%   25%   50%   75% 97.5%
## intercepto  -0.5 0.5  -1.5  -0.9  -0.6  -0.3   0.7
## betalwt      0.0 0.0   0.0   0.0   0.0   0.0   0.0
## betaht       1.4 0.4   0.5   1.1   1.4   1.7   2.2
## betasmoke    0.9 0.3   0.3   0.7   0.9   1.1   1.4
## betaui       0.6 0.3  -0.1   0.4   0.6   0.9   1.3
## betarace2    1.1 0.4   0.3   0.8   1.1   1.3   1.8
## betarace3    0.7 0.3   0.1   0.5   0.7   0.9   1.3
## deviance   201.2 3.6 196.3 198.5 200.4 203.1 209.8
##
## DIC info (using the rule, pD = Dbar-Dhat)
## pD = 6.2 and DIC = 207.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

As last thing, we can compare parameters values from different link functions for frquentist and Bayesian models.

```
a<-fitfinal$coefficients
c<-summary(bM)[1]
b<-c$statistics[,1]
c<- fitprobit$coefficients
d<-summary(bprobit)[1]
d<-d$statistics[,1]
e<-fitcloglog$coefficients
f<-mlgoutCloglog$summary[1:7,1]
data.frame(LogitFreq =a, LogitBayes=b, ProbitFreq=c, ProbitBayes=d, CloglogFreq=e, CloglogBayes=f, row.names = c("intercep
t","lwt","ht","smoke","ui","race2","race3"))
```

```
##             LogitFreq  LogitBayes   ProbitFreq ProbitBayes CloglogFreq
## intercept -0.03405073  0.05792750 -0.035050699 -0.01169644 -0.49750247
## lwt       -0.01596480 -0.01689866 -0.009520019 -0.00989165 -0.01194911
## ht         1.81049984  1.82185694  1.092199500  1.11655633  1.41292787
## smoke      1.13145231  1.14508330  0.690373916  0.70364283  0.86254660
## ui         0.87713522  0.87923220  0.531915100  0.53794561  0.66403422
## race2      1.34380710  1.33579156  0.804594304  0.82568885  1.10324532
## race3      0.88163212  0.87954865  0.525298253  0.53862680  0.68635915
##           CloglogBayes
## intercept  -0.54223964
## lwt        -0.01196109
## ht          1.36799820
## smoke       0.87647617
## ui          0.62838924
## race2       1.08374062
## race3       0.68026943
```