Exam

Bayesian statistics using JAGS and R

Omkar S Kulkarni

September 8, 2016

Contents

1 Exercise 1: 2

1.1 Let us visualize the distribution given 2

1.2 the slice sampler 2

1.3 simulate from the sampler and plot output 4

2 Exercise 2: 4

2.1 estimate a regression with lcavol as dependent variable (add intercept) . 7

2.2 Check convergence and interpret the results 11

2.3 Make a prediction for the average observation 11

2.4 Investigate and summarize the predictive distribution 13

3 Exercise 3: 14

3.1 estimate a logistic regression with churn as dependent variable 15

3.1.1 Robust logistic regression with JAGS 15

3.1.2 Comparison with normal logistic regression 16

3.2 Check convergence and interpret the results 17

3.3 redo the analysis with a probit model 17

3.4 what is the probability that $\beta_{recency} \leq 0$? 19

3.5 what is the probability that $\beta_{gender} > \beta_{lor}$? 19

3.6 Make a prediction for the average observation 19

4 Exercise 4: Bayesian quantile regression 21

4.1 Write an R function that estimates the parameters of the model proposed in the paper. 21

4.2 Do the simulation exercise in 5.1 Simulated data and compare your results with the results in the paper. 23

4.3 Simulate 200 observations from the following model: 25

4.4 Plot the data together with the OLS regression line. 26
4.5 Estimate the following quantiles q = {.05, .25, .5, .75, .95} and plot the regression lines based on the Bayes estimates on the same plot. 26

5 Exercise 5: 27
5.1 Calculate a frequentist t-test and ANOVA and interpret the results . . . 27
5.1.1 ANOVA 30
5.2 Calculate a Bayesian t-test and ANOVA 31
5.2.1 Hypothesis testing with HDI and ROPE 31

1 Exercise 1:

Create a slice sampler to sample from mixture of normals distribution given here:

```
rm(list=ls()) # remove all variables
cat("\014") # clear console
```

```
set.seed(1212)


dbinorm <- function(x, p1=.5, p2=1-p1, m1=-1, m2=2){
p1 * dnorm(x, m1) + p2 * dnorm(x, m2)
}
```

1.1 Let us visualize the distribution given

```
input <- seq(-4.5,+5.5,0.001)
distribution <- dbinorm(input)
plot(input, distribution,type = 'l', main = "given distribution")
```

```
# we observe a bimodal distribution with peaks at -1 and 2.
```

1.2 the slice sampler

```
#### Function for Slice sampler #####
Slice.sample.fn = function(N, StartPoint, stepsize){
# Initialize vectors
# Keep track of: samples, counters for steps and resamples
# Set initial time t
```

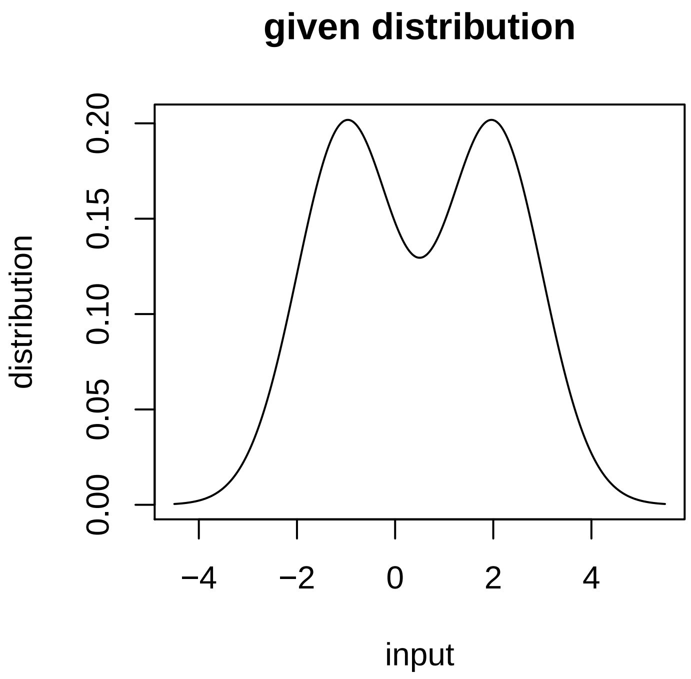


Figure 1: Density of the mixture distribution

```
  x = vector(length = N) # Samples
  count = 0
  count_lower= 0
  count_upper = 0
  resample =0

  # Starting point
  x[1]= StartPoint

#Actual sampling:
# 1. from a uniform distribution [0, P(theta|data)] at given theta -- sample
# 2. Find the boundaries of the distribution at the place of sample
# 3. Draw a sample from a uniform over the two boundaries
# and if the sample is outside the boundaries, resample
# 4. Repeat using new theta
  for (i in 1:(N-1)){

      temp = runif(1, 0, dbinorm(x[i])) # Draw 1 sample
      upper= lower = x[i] # Initialize boundaries
       #1
      while(dbinorm(upper) > temp){ # Determine the boundaries
      upper=  upper + stepsize   # add step size till while fails
      count_upper = count_upper + 1
      }
```

```
    #2
  # Two separate while loops so they can be determined independently
    while(dbinorm(lower) > temp){
      lower = lower - stepsize   # substract step size till while fails
      count_lower = count_lower + 1
    }

      ## now we have got the upper and lower boundaries
      #3

  # Draw a sample from a uniform over the boundaries
    x[i+1] = runif(1, lower, upper)

    while(dbinorm(x[i+1]) < temp){
      x[i+1] = runif(1, lower, upper) # Resample
      resample= resample + 1
      }
  }


  count=  max(count_upper, count_lower)
  X =list(samples = x, count = count, resample = resample)
  return(X)
}
```

1.3 simulate from the sampler and plot output

We now sample the distribution with our slice sampler in Figure 2. We plot the original distribution in blue. We already anticipate the start point and hence burn in is not considered, but it can be accomodated. Also, effect of different step size (which is 1 in our case) can be shown.

```
Output = Slice.sample.fn(N = 100000, StartPoint = 1, stepsize = 1)

hist(Output$samples, main = "", ylab = "Density",xlab="X", freq = FALSE,breaks = 75)
input <- seq(-4.5,+5.5,0.001)
distribution <- dbinorm(input)
lines(input, distribution, col = 'blue', lty = 4, lwd =2)
```

2 Exercise 2:

Load the prostate cancer dataset from the bayesQR R-package. These data come from a study that examined the correlation between the level of prostate specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. It is a data frame with 97 rows and 9 columns. Use JAGS to solve this exercise.

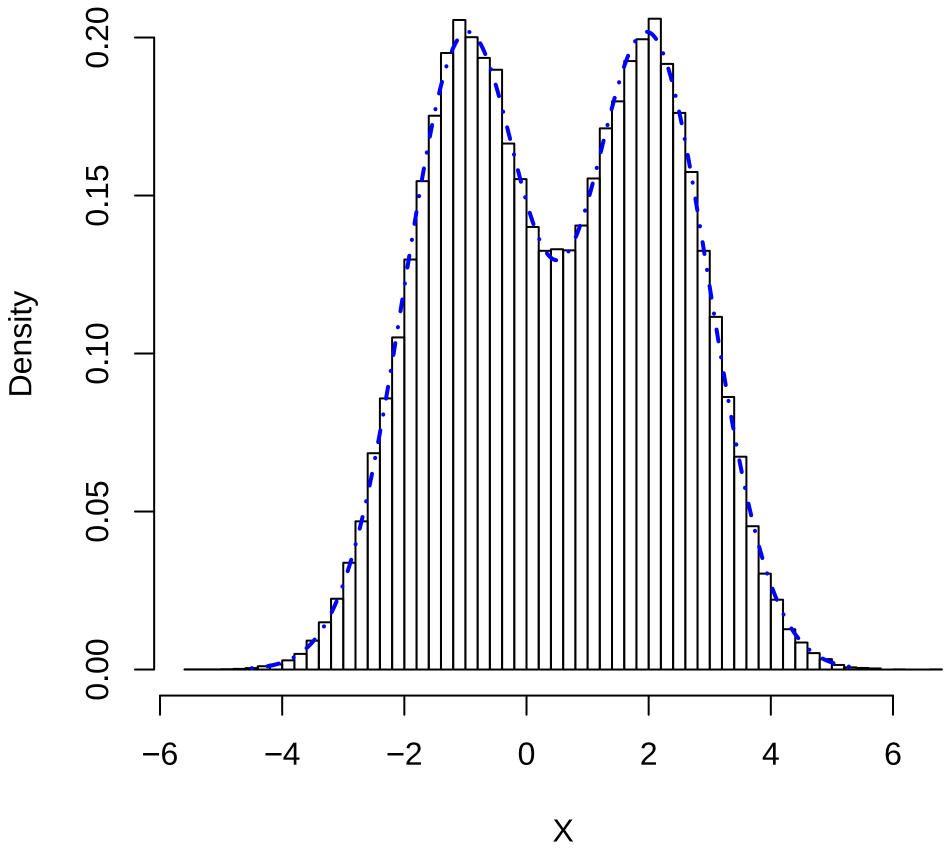


Figure 2: samples drawn from the slice sampler, blue line shows the original distribution, with 100000 samples and start point is 1

- estimate a regression with lcavol as dependent variable
- all other variables are predictors (also add an intercept)
- Check convergence and interpret the results
- Make a prediction for the average observation
- Investigate and summarize the predictive distribution

```
library(bayesQR)
data("Prostate")
#?Prostate
head(Prostate)
```

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
```

```
str(Prostate)
```

```
## 'data.frame':    97 obs. of  9 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : num  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: num  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : num  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```
summary(Prostate)
```

```
##      lcavol            lweight            age             lbph
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001
##  Mean   : 1.3500   Mean   :3.653   Mean   :63.87   Mean   : 0.1004
##  3rd Qu.: 2.1270   3rd Qu.:3.878   3rd Qu.:68.00   3rd Qu.: 1.5581
##  Max.   : 3.8210   Max.   :6.108   Max.   :79.00   Max.   : 2.3263
##       svi              lcp             gleason          pgg45
##  Min.   :0.0000   Min.   :-1.3863   Min.   :6.000   Min.   :  0.00
##  1st Qu.:0.0000   1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00
##  Median :0.0000   Median :-0.7985   Median :7.000   Median : 15.00
##  Mean   :0.2165   Mean   :-0.1794   Mean   :6.753   Mean   : 24.38
##  3rd Qu.:0.0000   3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00
##  Max.   :1.0000   Max.   : 2.9042   Max.   :9.000   Max.   :100.00
##       lpsa
##  Min.   :-0.4308
##  1st Qu.: 1.7317
##  Median : 2.5915
##  Mean   : 2.4784
##  3rd Qu.: 3.0564
##  Max.   : 5.5829
```

The data frame has the following components: *lcavol : log(cancer volume) lweight : log(prostate weight) age : age lbph : log(benign prostatic hyperplasia amount) svi : seminal vesicle invasion lcp : log(capsular penetration) gleason : Gleason score pgg45 : percentage Gleason scores 4 or 5 lpsa : log(prostate specific antigen)*

We input the data. Take all the predictors in 'x' and the independent variable 'lcavol'

```
# create the input and output in proper format
x = data.frame((Prostate[,-1]))
lcavol <- Prostate[,1]
```

We use **JAGS** for the linear regression :

$$E(Y|X_i) = \beta_0 + \Sigma(\beta_j x_j) + \epsilon_i$$

where $\epsilon\tilde{N}(0, \sigma^2)$; Once we get the output we compare it with classical linear regression (lm() - Fitting Linear Models).

2.1 estimate a regression with lcavol as dependent variable (add intercept)

```
# lets define the model
#The model can be stored either as a separate file or given inline as below.
# http://www4.stat.ncsu.edu/~reich/st590/code/regJAGS
modelJags <- "
      data {
          meanResponse <- mean(lcavol)
          nu <- length(lcavol) - 2
      }
      model {
      # Likelihood
          for (i in 1:length(lcavol)){
          lcavol[i] ~ dnorm(lcavol.hat[i], 1/sigma^2)
          ##  precision = 1/sigma^2
          lcavol.hat[i] <- intercept + b[1]*lweight[i] + b[2]*age[i]
          + b[3]*lbph[i] + b[4]*svi[i] + b[5]*lcp[i]
          + b[6]*gleason[i] + b[7]*pgg45[i]
          + b[8]*lpsa[i]
      }
      # Priors
      intercept ~ dnorm(meanResponse, 0.0001)
      for(i in 1:8) {
          betaMu[i] ~ dunif(-1000,1000)  # serve as a non-informative prior
          betaSigma[i] ~ dunif(0, 1000)
          b[i] ~ dt(betaMu[i], 1/betaSigma[i]^2, nu)

      }
      sigma ~ dunif(0, 100)
}"




require(rjags)
```

```
## Loading required package: rjags
```

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

```
## Linked to JAGS 4.2.0
```

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

```
regression.model <- jags.model(textConnection(modelJags), data = data.frame(lcavol, 
n.chains = 3, n.adapt = 500, quiet = FALSE)
```

```
## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 97
##    Unobserved stochastic nodes: 26
##    Total graph size: 1773
##
## Initializing model
```

```
#update(regression.model, 100)
```

We set the number of chains to 3 and the parameter n.adapt to 500, hence the algorithm will advance 500 steps (system throws away 500 samples as part of the adaptive sampling period for each chain) and then collect samples. We now run the chain with n.iter = 100000.

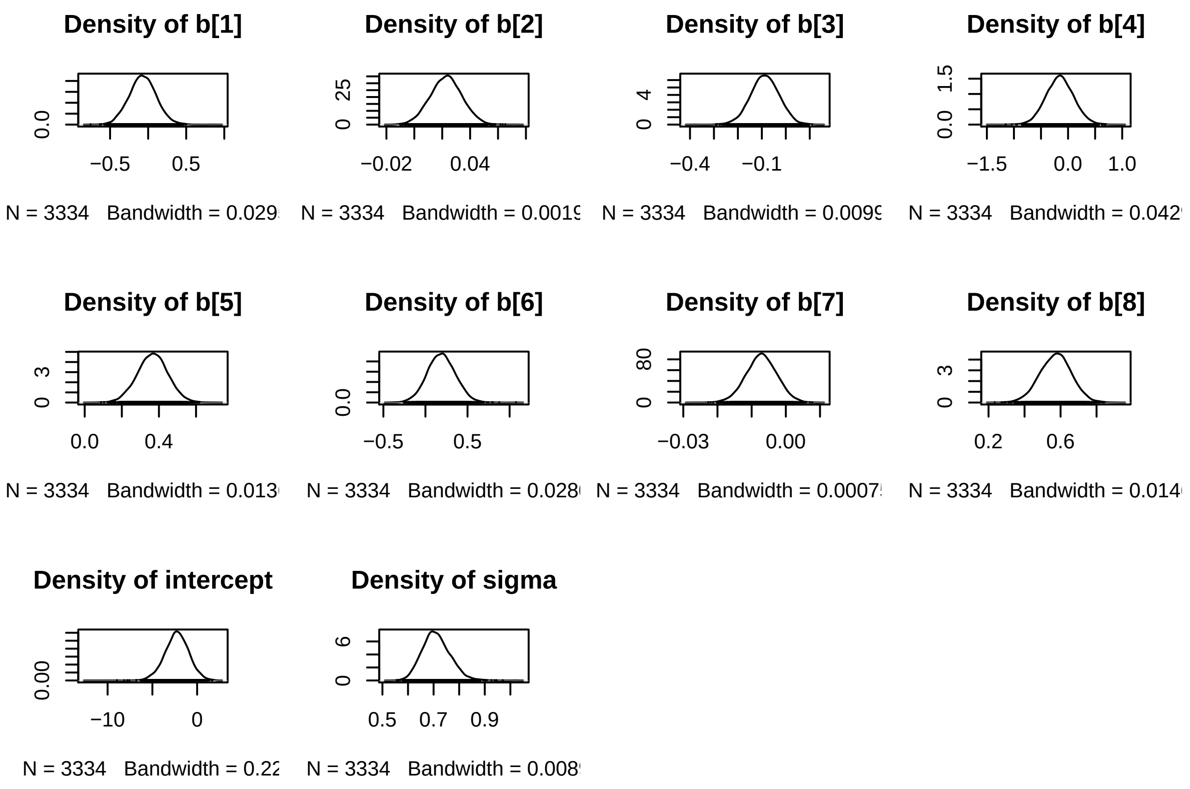
```
Output <- coda.samples(regression.model, variable.names = c("intercept", "b", "sigma"

summary(Output)
```

```
##
## Iterations = 530:100520
## Thinning interval = 30
## Number of chains = 3
## Sample size per chain = 3334
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean       SD  Naive SE Time-series SE
## b[1]      -0.070146 0.178058 1.780e-03      0.0041620
## b[2]       0.022844 0.011315 1.131e-04      0.0003049
## b[3]      -0.087197 0.059661 5.965e-04      0.0008745
## b[4]      -0.150580 0.255725 2.557e-03      0.0026872
## b[5]       0.366078 0.082453 8.244e-04      0.0008816
## b[6]       0.189949 0.166865 1.668e-03      0.0073634
## b[7]      -0.007129 0.004499 4.499e-05      0.0001210
## b[8]       0.572814 0.087392 8.738e-04      0.0010852
## intercept -2.272564 1.354508 1.354e-02      0.0629872
## sigma      0.710723 0.054603 5.460e-04      0.0006158
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%       50%       75%    97.5%
## b[1]      -0.4129990 -0.18844 -0.072306  0.047296 0.282455
## b[2]       0.0005059  0.01522  0.022912  0.030458 0.045016
## b[3]      -0.2063764 -0.12646 -0.087072 -0.047317 0.027447
## b[4]      -0.6391473 -0.32413 -0.152556  0.019067 0.351631
## b[5]       0.2034625  0.31155  0.366746  0.420566 0.528519
## b[6]      -0.1358198  0.07634  0.188723  0.301673 0.518325
## b[7]      -0.0160860 -0.01010 -0.007107 -0.004111 0.001676
## b[8]       0.3991348  0.51423  0.574639  0.631328 0.742360
## intercept -5.0356777 -3.13640 -2.251305 -1.381879 0.356604
## sigma      0.6134188  0.67356  0.707276  0.745189 0.825308
```

We generate the output samples of the intercept and the β coefficiets and σ parameters, plotting their density curves respectively.

```r
par(mfrow = c(3,4))
plot(Output, trace = FALSE, auto.layout = FALSE)
par(mfrow = c(1,1))
```



We now compare the results with that of the coefficients of ordinary linear regression. we expect them to be same.

```r
## OLS regression
fit.lm = lm(lcavol~., data = x)
coef(fit.lm)
```

```
##  (Intercept)      lweight          age         lbph          svi
## -2.260100746 -0.073166458  0.022736050 -0.087449206 -0.153591285
##          lcp      gleason        pgg45         lpsa
##  0.367299994  0.190758511 -0.007157575  0.572797361
```

```r
summary(Output)$statistic[,1:2]
```

```
##                   Mean          SD
## b[1]      -0.070145720 0.178057554
## b[2]       0.022843747 0.011314932
## b[3]      -0.087197042 0.059660736
## b[4]      -0.150580155 0.255724502
## b[5]       0.366078107 0.082452571
## b[6]         0.189949246 0.166865318
## b[7]        -0.007128998 0.004499242
## b[8]         0.572814213 0.087391665
## intercept -2.272563778 1.354508058
## sigma        0.710723404 0.054602761
```

Outcomes are close.

2.2 Check convergence and interpret the results

Assessing convergence is a much more nuanced process, as there are multiple ways to decide if the model has converged. What we want to determine is that our model has appropriately explored the parameter space for each parameter (through trace plots, traceplot function in the coda library), between and within chain variance (the gelman-rubin diagnostic, gelman.diag in the coda library), and auto-correlation in our chains (autocorr.plot in coda).

We want to ensure that the traceplots are well-mixed for all the parameters (for all chains) which is clear from figure 3, that Gelman Rubin diagnostic is < 1.10 for each parameter (general rule), and from figure 4 the shrink factor is close to 1. And that the chains are not too correlated (this will reduce the effective sample size in our chains), figure not shown.

```r
library(lattice)
xyplot(Output)
```

```r
par(mfrow = c(3,4))
gelman.plot(Output, auto.layout = FALSE)
par(mfrow = c(1,1))
```

2.3 Make a prediction for the average observation

We know that :

$$\hat{Y}|\mathbf{x} = \hat{B}_0 + \sum \hat{B}_j x_j = \hat{\mathbf{B}} \cdot [1, \mathbf{x}]$$

where $\mathbf{x}$ is vector of predictors, and $\hat{B}$ are random variables from which we have the Output. Hence,

We start with a vector of predictors (in this case for average observation) and $\hat{B}$ to calculate for the final prediction.

```r
X_avg =colMeans(Prostate[, -1])
X_avg
```

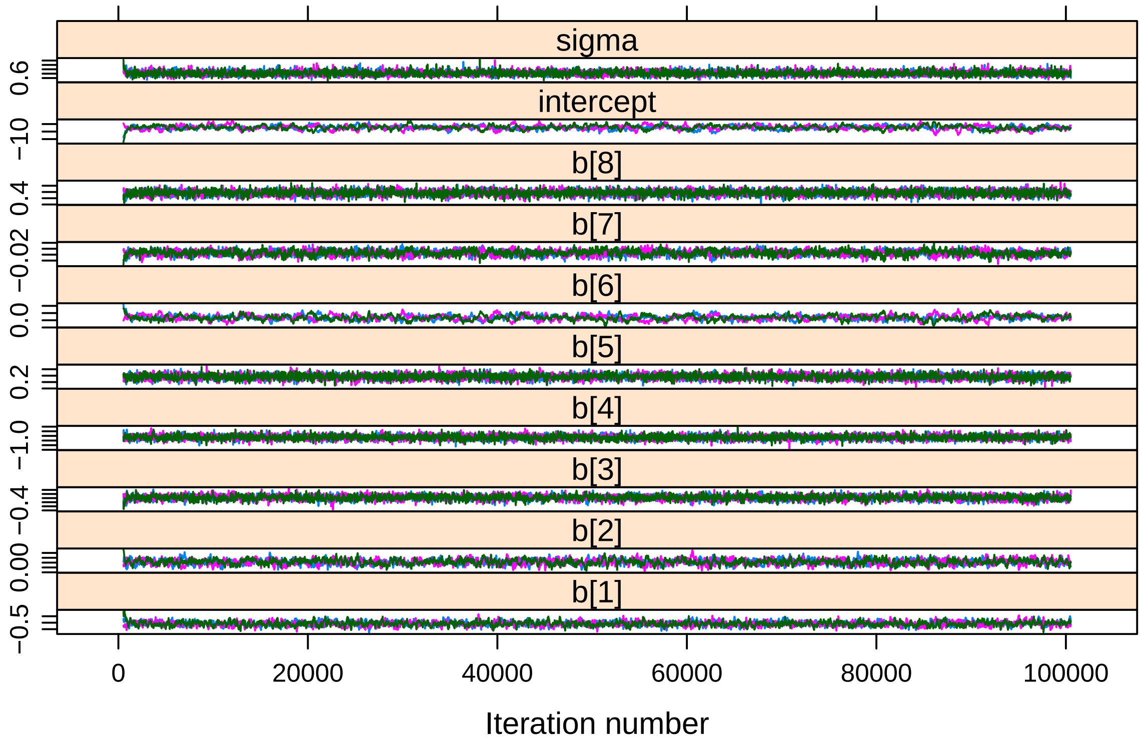


Figure 3: the three chains are overlapping for all parameters.

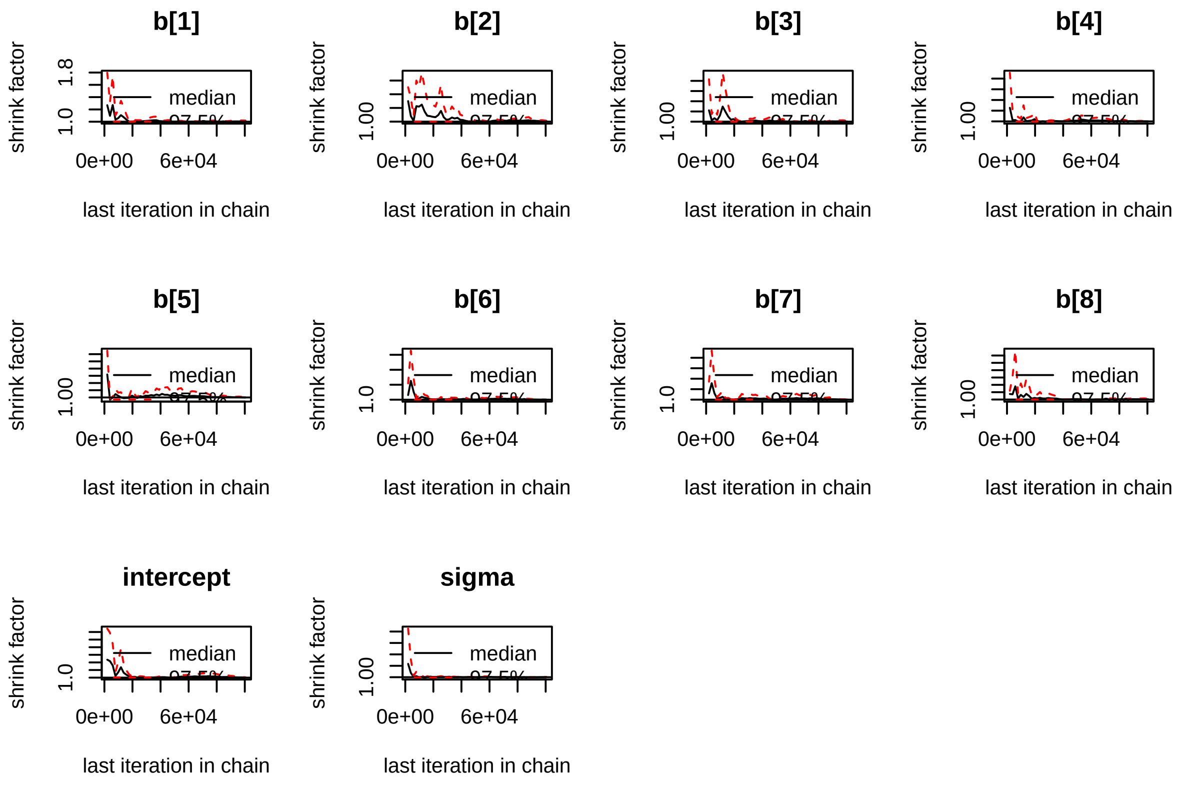


Figure 4: Gelman Plot, Srink factor for all parameters is <1.10

```
##    lweight        age       lbph        svi        lcp    gleason
##  3.6526864 63.8659794  0.1003556  0.2164948 -0.1793656  6.7525773
##      pgg45       lpsa
## 24.3814433  2.4783869
```

```
beta_Names <- c('intercept', 'b[1]', 'b[2]', 'b[3]', 'b[4]', 'b[5]', 'b[6]', 'b[7]',
Beta_hat = as.matrix(Output[[1]][,beta_Names])

round(Beta_hat[1:4,],2) # sample
```

```
##      intercept b[1] b[2]  b[3]  b[4] b[5] b[6]  b[7] b[8]
## [1,]     -8.95 0.29 0.02 -0.23  0.22 0.24 1.08 -0.02 0.43
## [2,]     -7.80 0.09 0.03 -0.12  0.64 0.22 0.88 -0.02 0.52
## [3,]     -7.58 0.05 0.05 -0.20 -0.55 0.41 0.75 -0.02 0.59
## [4,]     -8.15 0.22 0.04 -0.19  0.05 0.25 0.79 -0.02 0.56
```

```
Y_hat <- Beta_hat %*% append(X_avg, values = 1, after = 0)
```

Thus, the expected value of Y_hat is

```
mean(Y_hat)
```

```
## [1] 1.349738
```

And that in comparisomn to the predictions by OLS regression is :

```
predict(fit.lm, newdata = as.data.frame(t(X_avg)))[[1]]
```

```
## [1] 1.35001
```

Hence, the predicted mean of the average log(cancer volume) is 1.35

2.4 Investigate and summarize the predictive distribution

Now we plot the predictive distribution for the average observation in next figure.

```
library(BEST)
par(mfrow = c(1,1))
plotPost(Y_hat, xlab='lcavol')
```

```
predict(fit.lm, newdata = as.data.frame(t(X_avg)), interval = "prediction")
```

```
##       fit         lwr      upr
## 1 1.35001 -0.04780626 2.747825
```

And compare the Bayesian 95% Confidence interval in Figure with the classical linear regression prediction intervals.

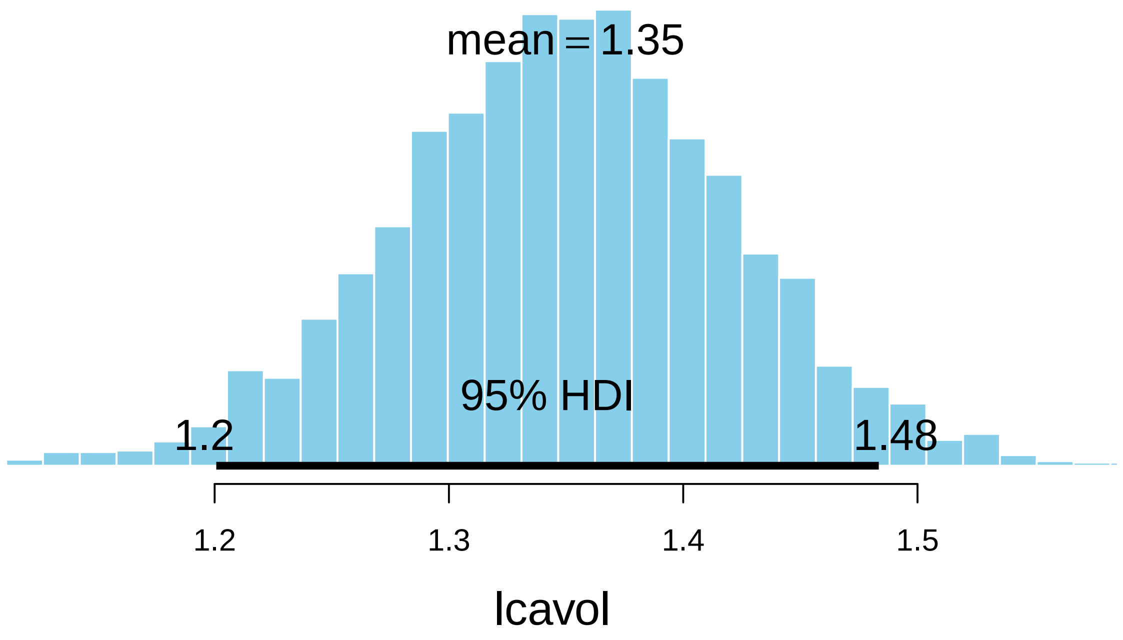


Figure 5: Posterior predictive distribution of lcavol for mean observations

3 Exercise 3:

Load the Churn dataset from the bayesQR R-package and read the documentation (e.g. ?Churn). Use JAGS to solve this exercise.

- churn : churn (yes/no)
- gender : gender of the customer (male = 1)
- soc : social class of the customer
- lor : length of relationship with the customer
- recency : number of days since last purchase

```
library(bayesQR)
data("Churn")
```

This dataset is a stratified random sample from all active customers (at the end of June 2006) of a European financial services company. The dependent variable in this dataset is the churn behavior of the customers in the period from July 1st until December 31th 2006. Here a churned customer is defined as someone who closed all his/her bank accounts with the company. Note that all predictor variables are standardized. This dataset is a small subset of the dataset used in Benoit and Van den Poel (2013). The dataset is structured as a dataframe with 400 observations and 5 variables.

```r
# proportion of churned customers
sum(Churn$churn)/nrow(Churn)
```

```
## [1] 0.5
```

```r
names(Churn)[3] <- 'soc'
head(Churn)
```

```
##       churn gender        soc         lor    recency
## 4632      0      1 -0.1156749 -1.08922097 -0.7213215
## 9695      0      0 -0.3134247  1.18298297  3.6344354
## 14160     0      1 -1.3254387 -0.84615637 -0.4275823
## 11016     0      1  1.9665148  0.08694165 -0.5356717
## 16286     0      1  1.4546917 -1.16664155 -0.6726400
## 11842     0      0 -0.1622042  0.49339968 -0.7700030
```

3.1 estimate a logistic regression with churn as dependent variable

3.1.1 Robust logistic regression with JAGS

According to classical logistic regression model the assumption is$churn \sim Bernoulli(\mu)$. and μ is modeled according to the following equation

$$logit(\mu) = log(\frac{\mu}{1-\mu}) = \beta_0 + \sum \beta_i x_i$$

which can be rewritten as

$$\mu = logit^{-1}(\beta_0 + \sum \beta_i x_i) = \frac{1}{1 + exp(-\beta_0 - \sum \beta_i x_i)}$$

We try to keep the model on similar lines to logistic regression.

```r
# http://stephenrho.github.io/rjags-model.html
# Rather than modelling P, which falls within [0,1], directly we model the log
#odds of success which spans [-inf, inf].


jags.logistic <- "
  data {
    out.r <- mean(churn)
    nu <- length(churn) - 2
  }
  model {
    # Likelihood

    for ( i in 1:length(churn) ) {
    logit(mu[i]) <- intercept + b[1] * gender[i] + b[2] * soc[i]
                              + b[3] * lor[i] + b[4] * recency[i]
    churn[i] ~ dbern(mu[i])
    }

    # Priors
    intercept ~ dt(out.r, 1/10^2, nu)
    for(i in 1:4) {
      bMu[i] ~ dunif(-1000, 1000)
      bSigma[i] ~ dunif(0, 1000)
      b[i] ~ dt(bMu[i], 1/bSigma[i]^2, nu)
    }
}"
```

Again we initialize the sampler with n.chains = 3 and n.adapt = 1000 (burn in period).And generate the samples.

```
library(rjags)
model.logistic <- jags.model(textConnection(jags.logistic), data = Churn,
                         n.chains = 3, n.adapt = 1000, quiet = TRUE)

Logit.output <- coda.samples(model.logistic, variable.names = c("intercept", "b"), n.
```

And below are the results of of the logit.

```
summary(Logit.output)$statistic[,1:2]
```

```
##                    Mean         SD
## b[1]       -0.06886179 0.20929731
## b[2]        0.01886216 0.09709814
## b[3]       -0.65618654 0.13963908
## b[4]        0.52070866 0.11534499
## intercept -0.08287881 0.15263012
```

3.1.2 Comparison with normal logistic regression

```
fit.logit <- glm(churn ~ ., data = Churn, family = binomial(link = "logit"))
coef(fit.logit)
```

```
## (Intercept)      gender         soc         lor     recency
## -0.08268175 -0.06512772  0.01850613 -0.64341795  0.50852148
```

Hence, on comparison coefficients obtained by Bayesian methods and that by normal GLM (logit link) are indeed very close.

3.2 Check convergence and interpret the results

Assessing convergence is a much more nuanced process, as there are multiple ways to decide if the model has converged. What we want to determine is that our model has appropriately explored the parameter space for each parameter (through trace plots, traceplot function in the coda library), between and within chain variance (the gelman-rubin diagnostic, gelman.diag in the coda library), and auto-correlation in our chains (autocorr.plot in coda).

```
library(lattice)
xyplot(Logit.output)
```

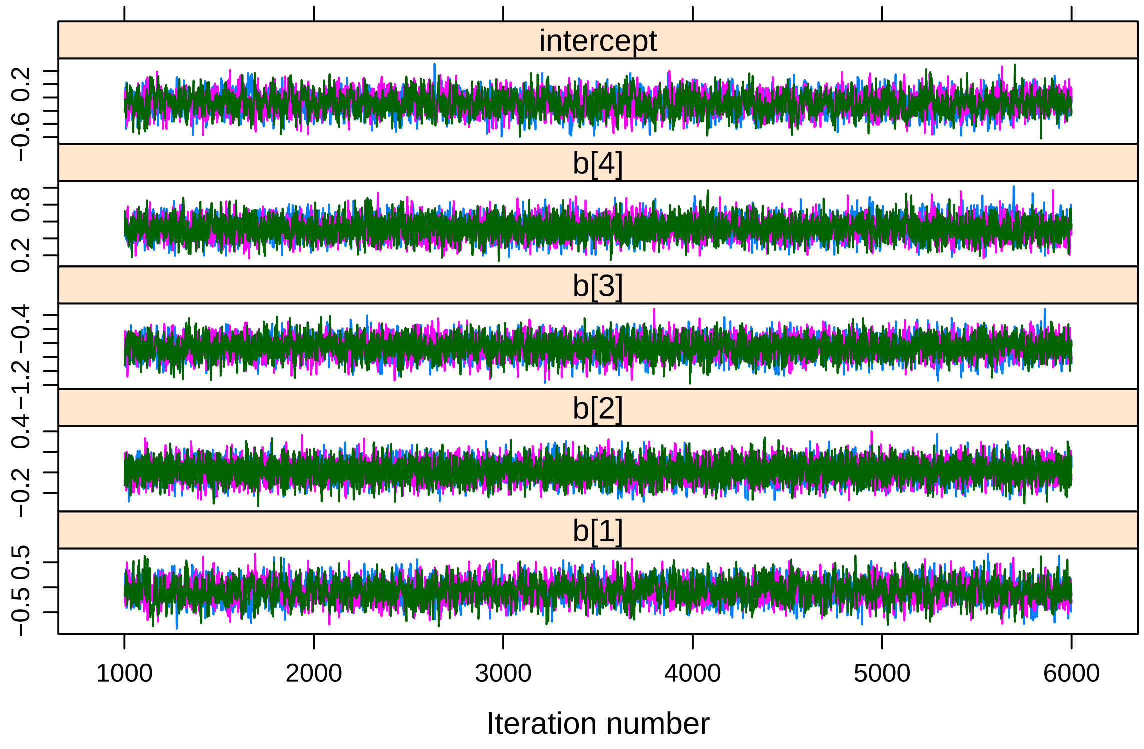


Figure 6: the three chains are overlapping for all parameters.

```
par(mfrow = c(2,3))
gelman.plot(Logit.output, auto.layout = FALSE)
par(mfrow = c(1,1))
```

3.3 redo the analysis with a probit model

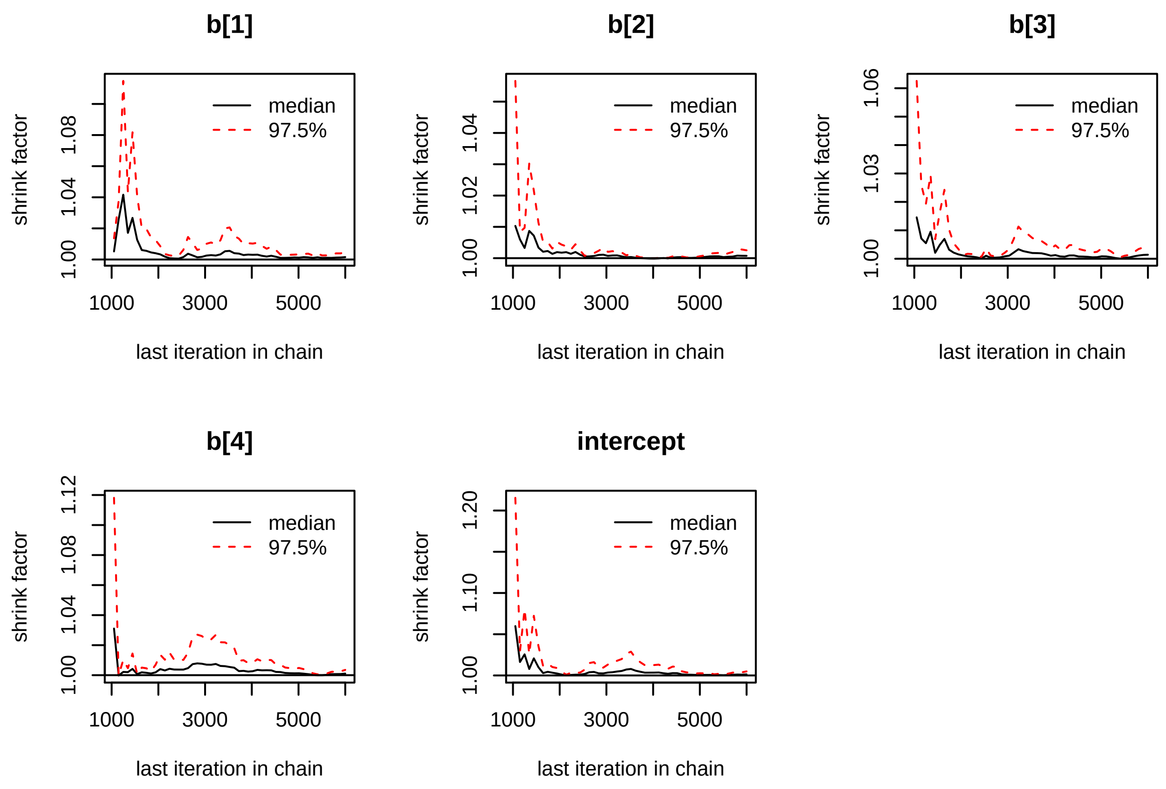


Figure 7: Gelman Plot, Srink factor for all parameters is <1.10

```r
# grep, grepl, regexpr, gregexpr and regexec search for
# matches to argument pattern within each element of a
# character vector: they differ in the format of and amount
# of detail in the results.

# sub and gsub perform replacement of the first and all
# matches respectively.
jags.probit <- gsub("logit", "probit", jags.logistic)

model.probit <- jags.model(textConnection(jags.probit), data = Churn,
                    n.chains = 3, n.adapt = 1000, quiet = TRUE)

Probit.output <- coda.samples(model.probit, variable.names = c("intercept", "b"),
                         n.iter = 10000, thin = 2)

summary(Probit.output)$statistic[,1:2]
```

```
##                  Mean         SD
## b[1]      -0.04102486 0.12818874
## b[2]       0.01474893 0.05926608
## b[3]      -0.39496459 0.08259158
## b[4]       0.31465873 0.06729850
## intercept -0.04809157 0.09290977
```

As previously we compare the Bayesian results to the normal probit regression.

```
## in the link function we set it to "probit"
fit.probit <- glm(churn ~ ., data = Churn, family = binomial(link = "probit"))
coef(fit.probit)
```

```
## (Intercept)      gender         soc         lor     recency
## -0.04922907 -0.04027200  0.01382450 -0.39257008  0.31128296
```

And again a good match. It was expected as there is no much difference in the 'working' of Probit and Logit. Logit has better interpretation than probit. Logistic regression can be interpreted as modeling log odds.

3.4 what is the probability that $\beta_{recency} \leq 0$?

We base our further inference based on the probit model. From the samples of the posterior distribution we can infere.We can estimate the probability that $\beta_{recency} \leq 0$ by looking at number/proportion of samples that are less than or equal to 0. And from the figure below we do not see any samples less than 0. And we can say $P(\beta_{recency} \leq 0) \approx 0$.

```
library(BEST)
plotPost(Probit.output[[1]][,"b[4]"], compVal = 0 )
```

3.5 what is the probability that $\beta_{gender} > \beta_{lor}$?

In the similar way we calculate the posterior probability $\beta_{gender} > \beta_{lor}$. Rather we can look at $\beta_{gender} - \beta_{lor} > 0$. And we plot histogram of the differences.

```
library(BEST)
plotPost(Probit.output[[1]][,"b[1]"] - Probit.output[[1]][,"b[3]"],compVal = 0)
```

From the figure we can say $P(\beta_{gender} - \beta_{lor} > 0) = P(\beta_{gender} > \beta_{lor}) \approx 0.985$.

3.6 Make a prediction for the average observation

```
X_avg = colMeans(Churn[,-1])
head(X_avg)
```

```
##      gender         soc         lor     recency
##  0.49500000 -0.05215644 -0.03437010  0.14454917
```

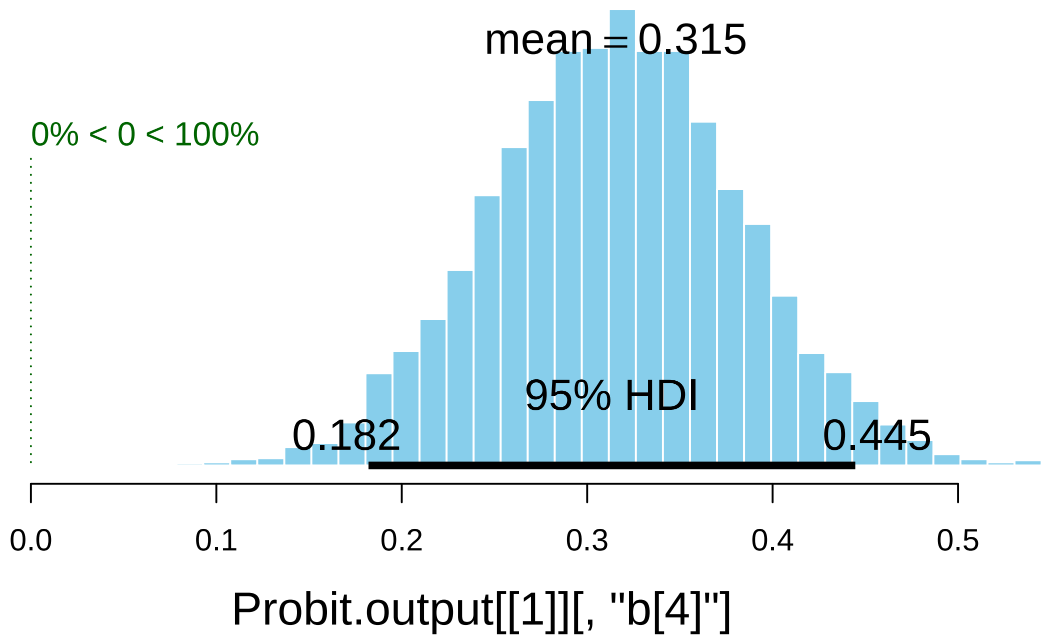


Figure 8: density plot

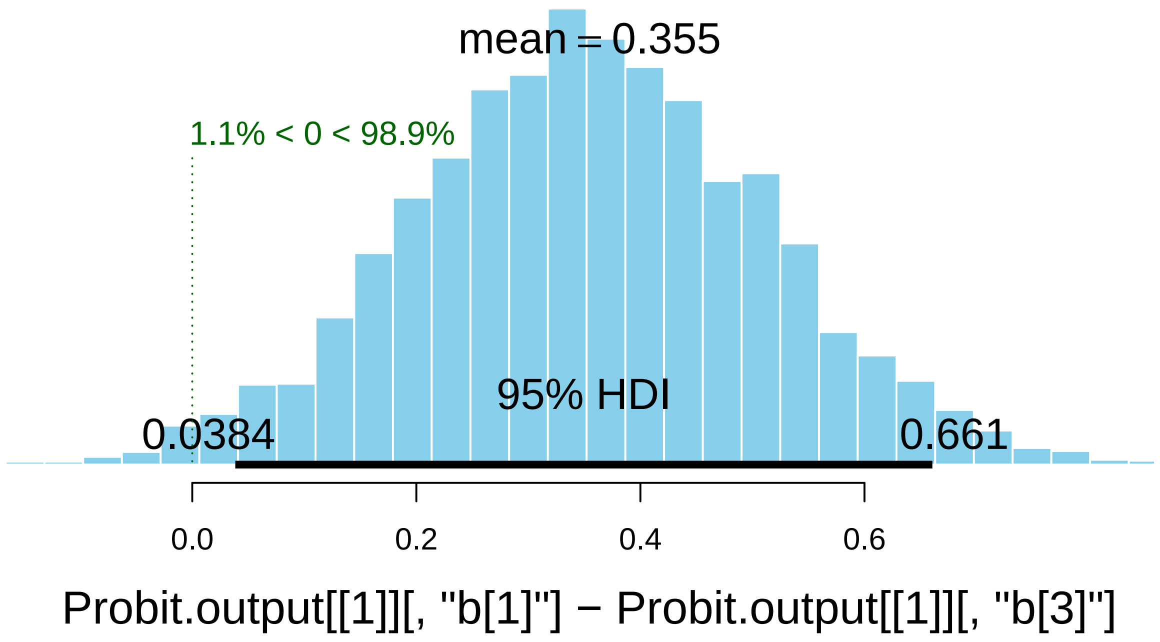


Figure 9: Density plot of Beta_gender - beta_lor

```r
X_avg[1] = 0
```

As gender=0.495 does not make sense, because it is a categorical variable we set it to 1, to represent the males. Lets predict now(As we did in the previous question)

```r
beta_Names <- c('intercept', 'b[1]', 'b[2]', 'b[3]', 'b[4]')
Beta_hat <- as.matrix(Probit.output[[1]][,beta_Names])

round(Beta_hat[1:4,],2) # sample
```

```
##      intercept  b[1]  b[2]  b[3] b[4]
## [1,]     -0.05  0.02 -0.03 -0.28 0.18
## [2,]     -0.04 -0.03  0.01 -0.31 0.30
## [3,]     -0.23  0.31  0.13 -0.32 0.40
## [4,]     -0.05 -0.03 -0.08 -0.40 0.24
```

```r
Y_hat <- Beta_hat %*% append(X_avg, values = 1, after = 0)
```

probability of churning for the average customer is : 0.50 as shown in the next figure.

```r
library(BEST)
plotPost(pnorm(Y_hat), xlab='Churn probability')
```

4 Exercise 4: Bayesian quantile regression

For this exercise you have to read the paper. Yu, K and Moyeed, R (2001). Bayesian quantile regression, Statistics & Probability Letters, 54(4), 437{447.

4.1 Write an R function that estimates the parameters of the model proposed in the paper.

the Metropolis algorithm with normal proposal density should work fine (as well as the slice sampler).

According the paper, the loss function is given by

$$\rho_p(u) = \frac{|u| + (2p-1)u}{2}$$

```r
Loss.Function = function(vec,p){
n  = length(vec)
rho.p= vector(length = n)
for (i in 1:n){
```

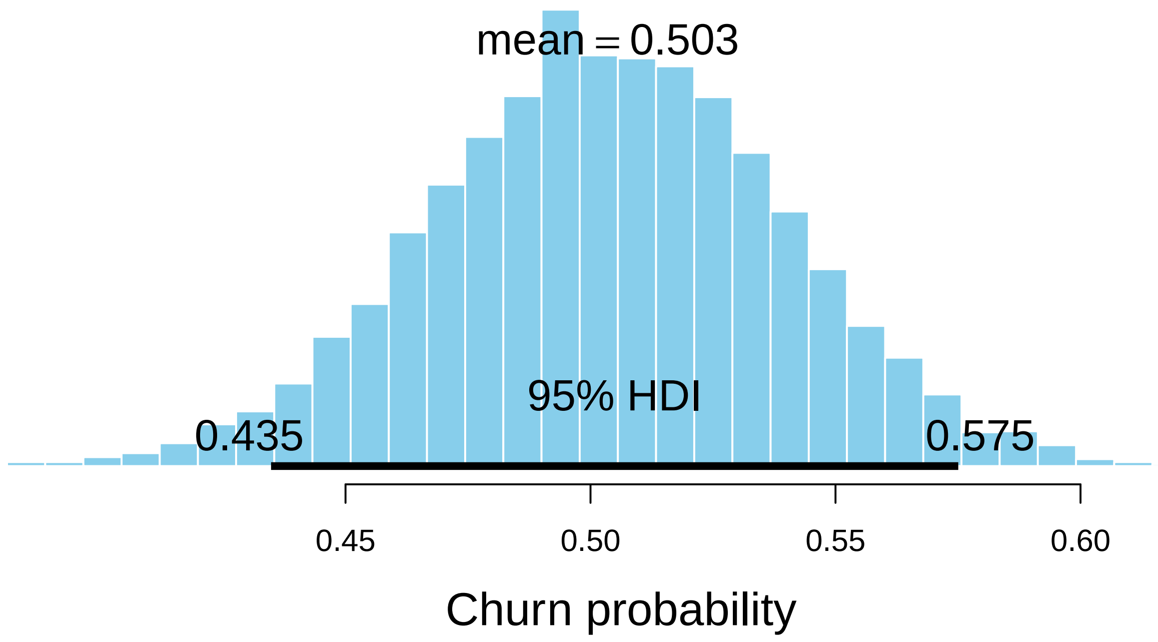


Figure 10: predictive distribution of churn probability for mean observation

```
      rho.p[i] =  (abs(vec[i]) + (2*p - 1)*vec[i]) * 0.5
    }

return(rho.p)
}
```

In the mentioned paper the likelihood is given by equation (6).

$$L(Y|\beta) = p^n(1-p)^n exp(-\sum(\rho_p(y_i - x_i\beta))$$

Or we can use the log likelihood to simplify the numerical calculations.

$$log(L(Y|\beta)) = nlog(p) + nlog(1-p) - \sum(\rho_p(y_i - x_i\beta)$$

```
log.likelihood = function( beta,y, x, p){
# Log likelihood
n = length(y)
log.likelihood  =  n*log(p) + n*log(1-p) - sum(Loss.Function((y - x*beta),p))

#likelihood  =  p^n * (1-p)^n * exp(- sum(Loss.Function((y - x*beta),p)))
```

```r
return(log.likelihood)
}
```

And for the prior as mentioned in the section 4 of the paper using improper uniform for the prior of β results in the proper joint posterior distribution. Its proof is shown in Appendix A of the mentioned paper.

```r
prior <- function(beta){
  return (1)
}
```

```r
Metropolis = function(N, stepsize, startpoint,y,x,p){
  v = vector(length = N)
  reject = 0
  v[1] = startpoint
  llikelihood = log.likelihood(v[1], y,x,p)

  for(i in 1:(N-1)){
    current = v[i]

    ## normal proposal density as mentioned
    proposal = rnorm(1,v[i],stepsize)

    ## acceptance probability
    llikelihood.proportion = log.likelihood(proposal,y,x,p)
    ll.diff =  llikelihood.proportion - llikelihood

    if(ll.diff >= log(runif(1))){
      v[i+1] = proposal
      llikelihood = llikelihood.proportion ## replace with new ll
    }else{
    v[i+1] = current
    reject = reject + 1 ## increase counter

    }
  }
  dist.output = list(samples = v, reject.rate=reject/N)
  return(dist.output)
}
```

4.2 Do the simulation exercise in 5.1 Simulated data and compare your results with the results in the paper.

The simulated data : generate n = 100 observations from the model

$$Y_i = \mu + \epsilon_i, i = 1, 2, ..., n$$

assuming that $\mu = 5.0$ and ϵ_i $N(0,1)$ for all $i = 1, 2, ..., n$

```
n=100
mu = 5
sd = 1
Y <- 5.0 + rnorm(n = n, mean = mu, sd = sd)
```

```
Metro.05 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =seq(from = 1, to = 1, length.out = n) , p = 0.05)
Metro.25 =  Metropolis(5000, stepsize = 5, startpoint = 3,
y = Y, x =seq(from = 1, to = 1, length.out = n) , p = 0.25)
Metro.75 =  Metropolis(5000, stepsize = 5, startpoint = 3,
y = Y, x =seq(from = 1, to = 1, length.out = n) , p = 0.75)
Metro.95 =  Metropolis(5000, stepsize = 5, startpoint = 3,
y = Y, x =seq(from = 1, to = 1, length.out = n) , p = 0.95)

par(mfrow=c(2,2))
plot(Metro.05$samples[], type = 'l', xlab = "p=0.05")
plot(Metro.25$samples[], type = 'l', xlab = "p=0.25")
plot(Metro.75$samples[], type = 'l', xlab = "p=0.75")
plot(Metro.95$samples[], type = 'l', xlab = "p=0.95")
```

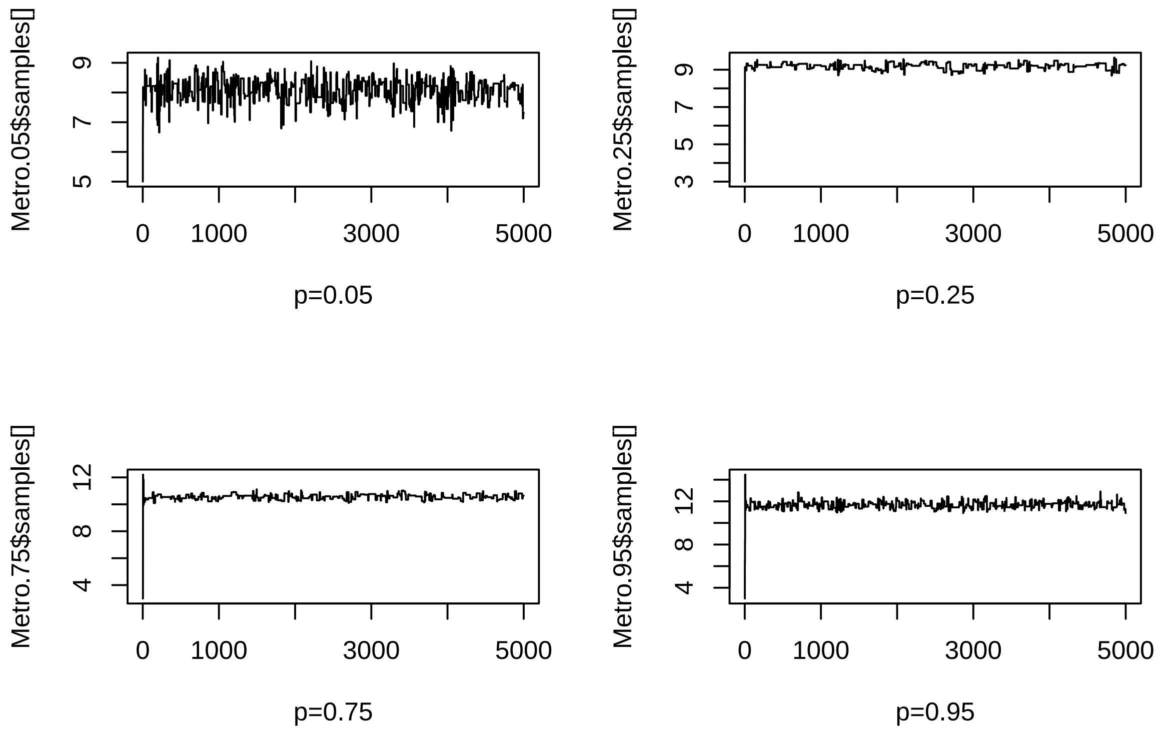


Figure 11: reach convergence withing few samples.

```r
par(mfrow = c(1,1))
```

From the trace plots, it is seen that convergence is achieved within few samples. Now the histograms :

```r
par(mfrow=c(2,2))
hist(Metro.05$samples, plot = TRUE, freq = TRUE)
hist(Metro.25$samples, plot = TRUE, freq = TRUE, breaks = 50)
hist(Metro.75$samples, plot = TRUE, freq = TRUE, breaks = 50)
hist(Metro.95$samples, plot = TRUE, freq = TRUE, breaks = 50)
```

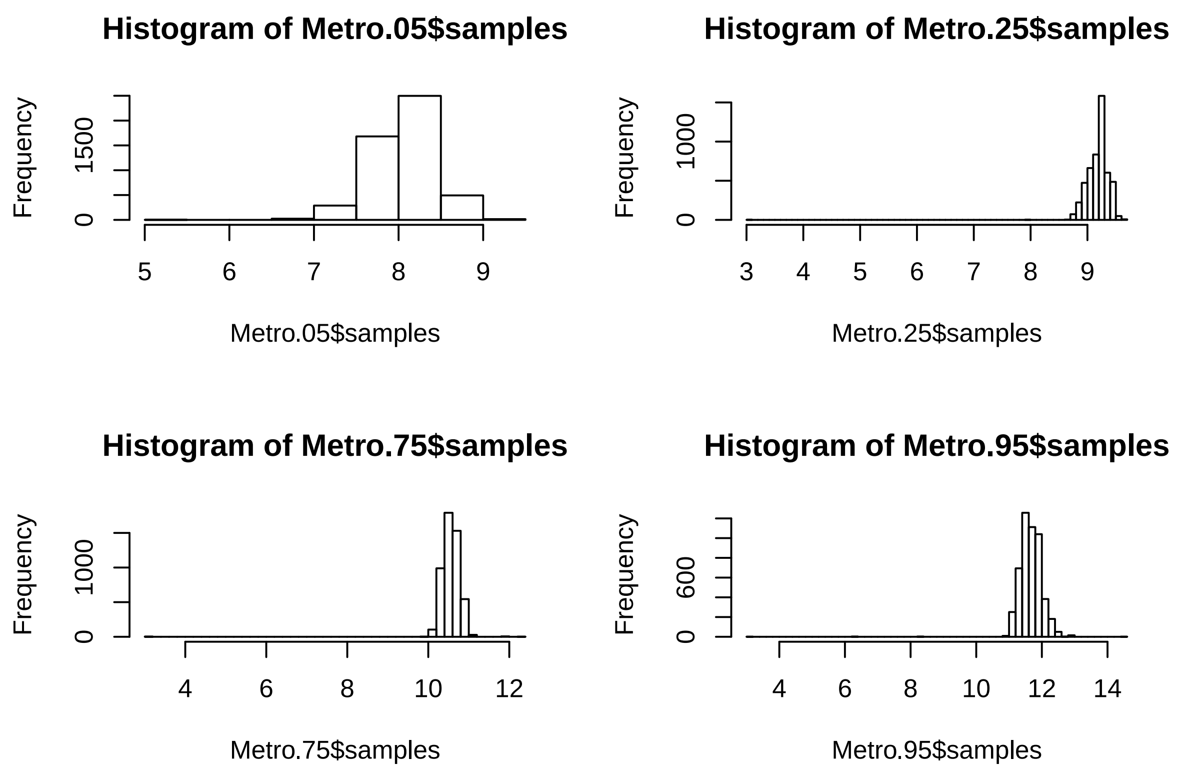


Figure 12: histograms of MCMC samples

```r
par(mfrow = c(1,1))
```

Clearly, only the shape of the histograms is 'similar' to figure 1 of the paper, however, the means and overall position is different. (couldnt debug this one, tried with likelihood too and not the log likelihood.)

4.3 Simulate 200 observations from the following model:

$$y = x\beta + \epsilon$$

with $\beta = 2\ x\ U(0, 10)$ and $\epsilon\ N(0, 0.6x)$

```r
N <- 200
X <- runif(n = N, min = 0, max = 10)
beta <- 2
error <- rnorm(200, 0, 0.6*X)
Y <- X * beta + error
```

4.4 Plot the data together with the OLS regression line.

Estimate ordinary least square regression, and plot it with data.

```r
fit.ols = lm(Y~X)

plot(X,Y)
abline(a = fit.ols$coefficients[1], b = fit.ols$coefficients[2], lwd=2)
```

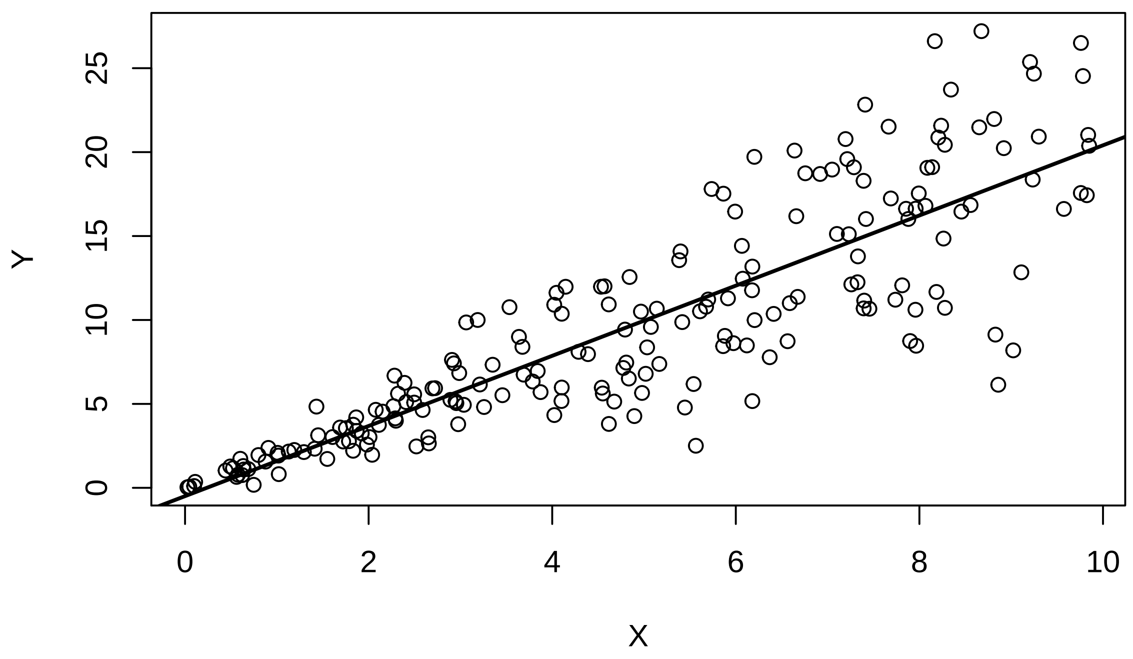


Figure 13: data together with the OLS regression line.

4.5 Estimate the following quantiles q = {.05, .25, .5, .75, .95} and plot the regression lines based on the Bayes estimates on the same plot.

We can use the same metropolis sampler to estimate the slope coefficients for q = {.05, .25, .5, .75, .95}.

```r
metro2.05 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =X , p = 0.05)

metro2.25 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =X , p = 0.25)

metro2.50 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =X , p = 0.50)

metro2.75 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =X , p = 0.75)

metro2.95 =  Metropolis(5000, stepsize = 5, startpoint = 5,
y = Y, x =X , p = 0.95)

slope.coefficients05 = mean(metro2.05$samples)
slope.coefficients25 = mean(metro2.25$samples)
slope.coefficients50 = mean(metro2.50$samples)
slope.coefficients75 = mean(metro2.75$samples)
slope.coefficients95 = mean(metro2.95$samples)

plot(X,Y)
abline(a = fit.ols$coefficients[1], b = fit.ols$coefficients[2], lwd=2)

intercept.ols = fit.ols$coefficients[1]
abline(a=intercept.ols, b = slope.coefficients05, col = 2)
abline(a=intercept.ols, b = slope.coefficients25, col = 3)
abline(a=intercept.ols, b = slope.coefficients50, col = 4)
abline(a=intercept.ols, b = slope.coefficients75, col = 5)
abline(a=intercept.ols, b = slope.coefficients95, col = 6)
```

5 Exercise 5:

The dataset ‘falcon data.RData’ was collected about Belgian falcons. The dataset contains information about the mass (in grams), the region (two levels) and the population (three levels) of the falcons.

The goal is to compare frequentist hypothesis testing and Bayesian hypothesis testing, using the falcon dataset.

5.1 Calculate a frequentist t-test and ANOVA and interpret the results

Let us start with the frequentist t-test and compare weights between two different regions.

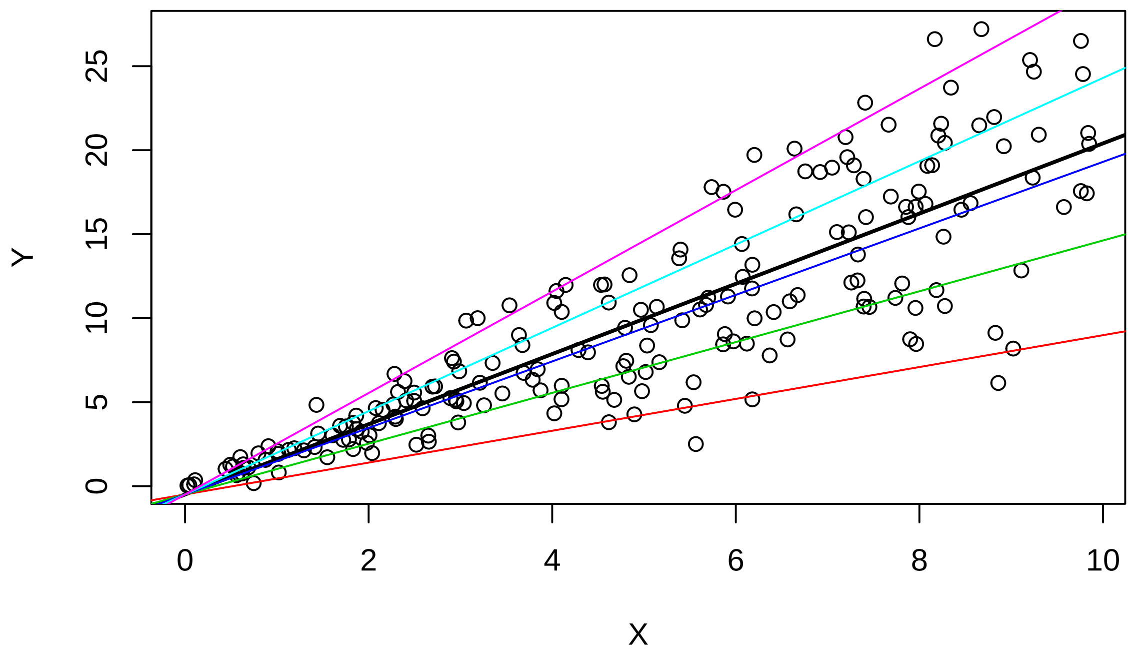


Figure 14: plot the regression lines based on the Bayes estimates

```r
#setwd("C:/Users/Omkar/OneDrive/Bayesian Statistics/Exam_Omkar")
load("falcon_data.RData")

# Visualize both weight in ddifferent regions.
boxplot(falcon_data$mass~falcon_data$region, xlab="region", ylab="Weight of falcon")
```

```r
boxplot(falcon_data$mass~falcon_data$population, xlab="Population", ylab="Weight of f
```

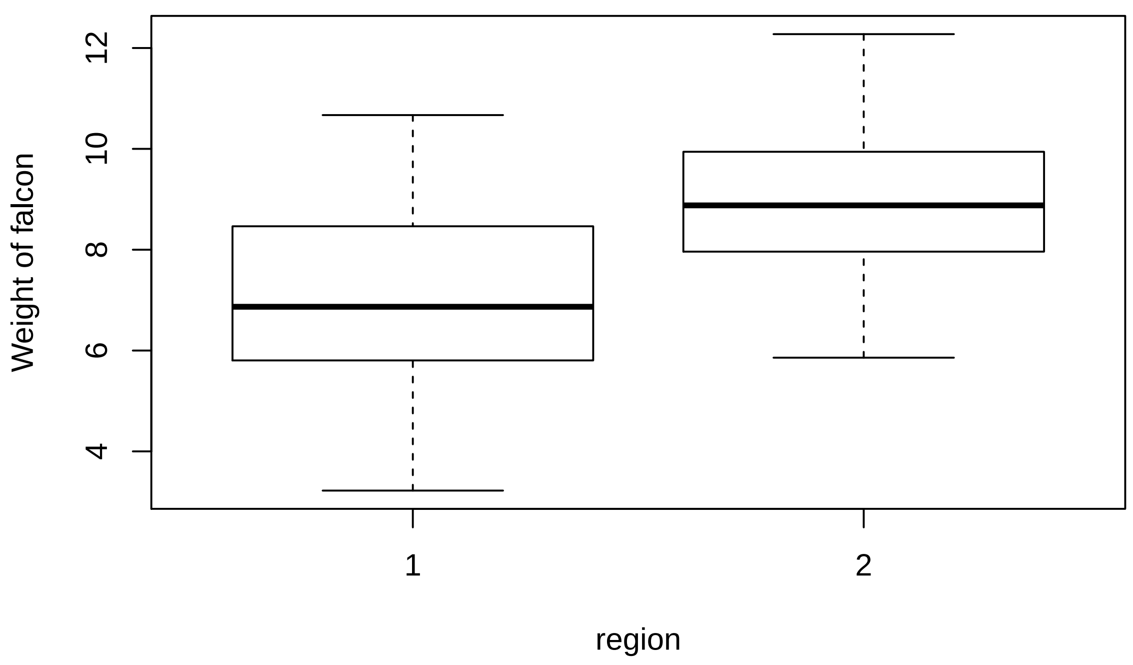
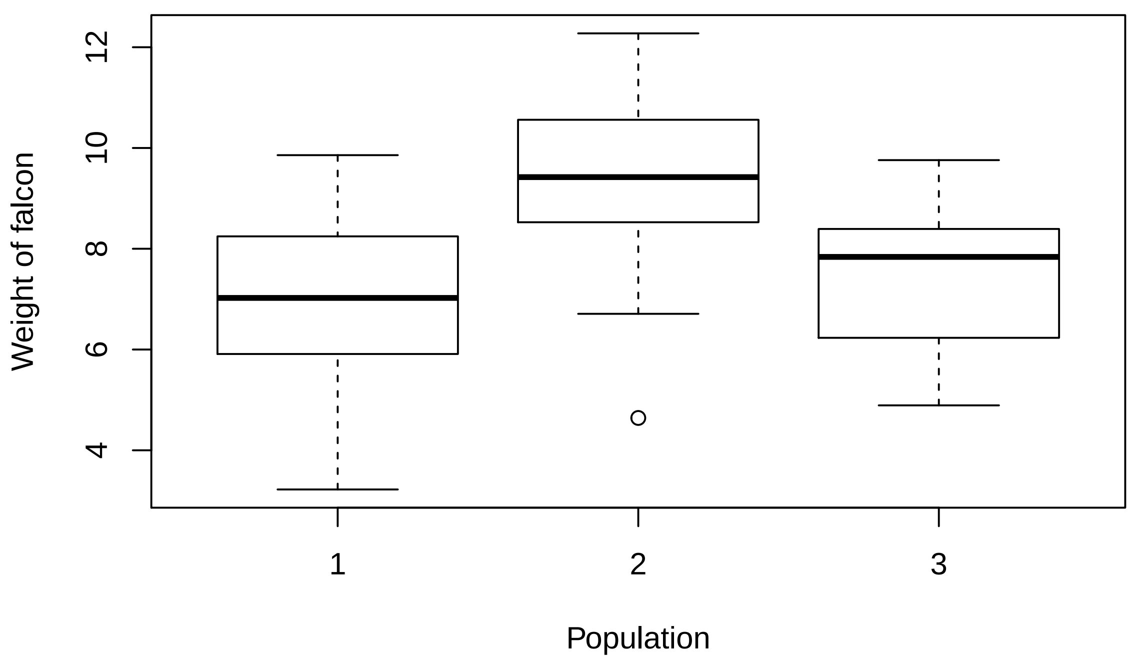


Figure 15: Boxplots of weight region wise.



A frequentist t-test tests the null hypothesis if the mean falcon weights in the two groups are the same:

```
t.test(mass~region, data = falcon_data)$p.value
```

```
## [1] 4.022563e-08
```

P value < 0.001, therefore we have enough evidence to reject null hypothesis on 5% significance level. Hence, a frequentist t-test accepts the alternative hypothesis that the group means differ.

5.1.1 ANOVA

The analysis of variance (ANOVA) generalizes the two-sample t-test to compare any number of means. ANOVA is based on the F statistic which under $H_0 : \mu_1 = \mu_2 = \cdots = \mu_G$:

Below are the results of ANOVA tests.

```
summary(aov(mass~region, data=falcon_data))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)    
## region       1  103.1  103.13   37.02 2.27e-08 ***
## Residuals   98  273.0    2.79                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(mass~population, data=falcon_data))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)  
## population   1   13.0  13.047   3.522 0.0635 .
## Residuals   98  363.1   3.705                 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(mass~population+region, data=falcon_data))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)    
## population   1  13.05   13.05   4.821   0.0305 *  
## region       1 100.55  100.55  37.151 2.22e-08 ***
## Residuals   97 262.52    2.71                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As expected, the Null hypothesis is rejected for all groups and subgroups.

5.2 Calculate a Bayesian t-test and ANOVA

To test the same hypothesis, we use linear model in Bayesian framework. Bayesian analysis assumes the t-distribution as a convenient (robust) descriptive model of data (with outliers), and not as the sampling distribution. The default priors taken are minimally informative: normal priors with large standard deviation for (μ), broad uniform priors for (σ), and a shifted-exponential prior for (ν).A convenient way to sample from this Bayesian $\mu_1 - \mu_2$ distribution is with the BEST package.

```
library(BEST)
mass_Region <-  with(falcon_data, BESTmcmc(mass[region==1], mass[region==2]))
mass_Pop1vs2 <- with(falcon_data, BESTmcmc(mass[population==1], mass[population==2]))
mass_Pop1vs3 <- with(falcon_data, BESTmcmc(mass[population==1], mass[population==3]))
mass_Pop2vs3 <- with(falcon_data, BESTmcmc(mass[population==2], mass[population==3]))
```

5.2.1 Hypothesis testing with HDI and ROPE

Bayesian hypothesis testing is done with ROPE and HDI. A region of practical equivalence (ROPE) indicates a small range of parameter values that are considered to be practically equivalent to the null value for purposes of the particular application. A parameter value is declared to be not credible, or rejected, if its entire ROPE lies outside the 95% highest density interval of the posterior distribution of that parameter, or vice versa accepted.

We now plot the distributions of $\mu_1 - \mu_2$ in below figures.

```
par(mfrow = c(2,2))
plot(mass_Region, main='Region', ROPE=c(-0.05, 0.05))
plot(mass_Pop1vs2, main='Population 1 vs 2', ROPE=c(-0.05, 0.05))
plot(mass_Pop1vs3, main='Population 1 vs 3', ROPE=c(-0.05, 0.05))
plot(mass_Pop2vs3, main='Population 2 vs 3', ROPE=c(-0.05, 0.05))
```

```
par(mfrow = c(1,1))
```

In case of Population 1 vs 3 we can not say that we accept the H_0 for practical purposes since the 95%HDI is not within the ROPE. Thus there is no concrete decision.

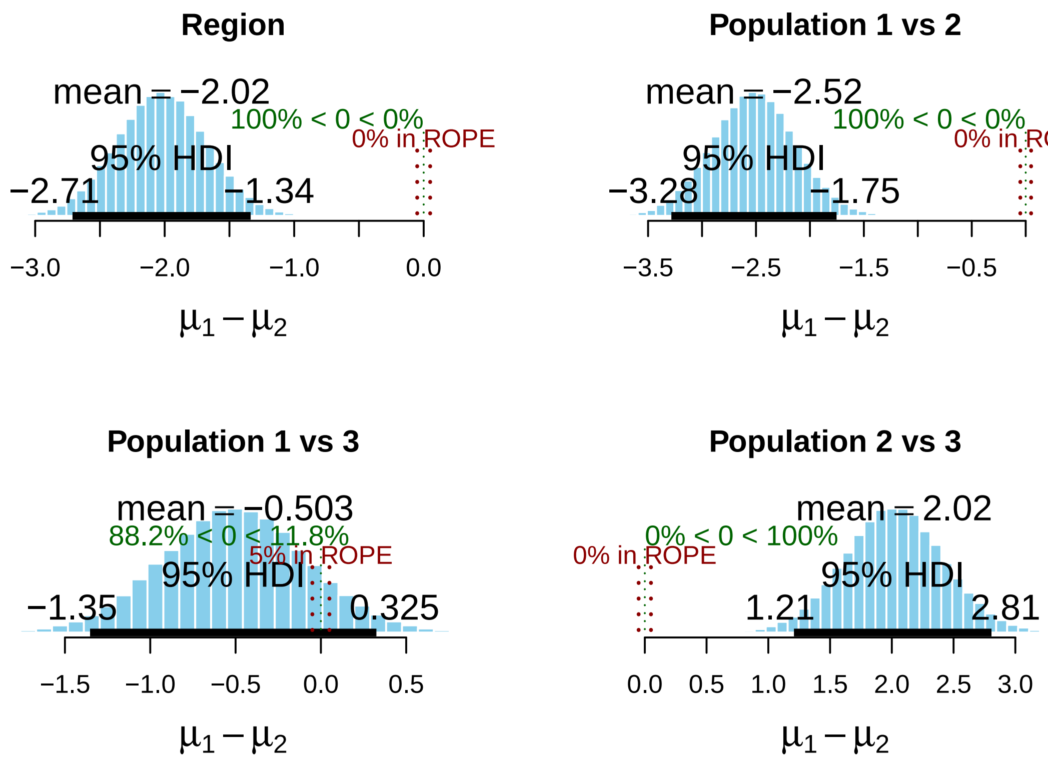


Figure 16: Bayesian two sample t-test.