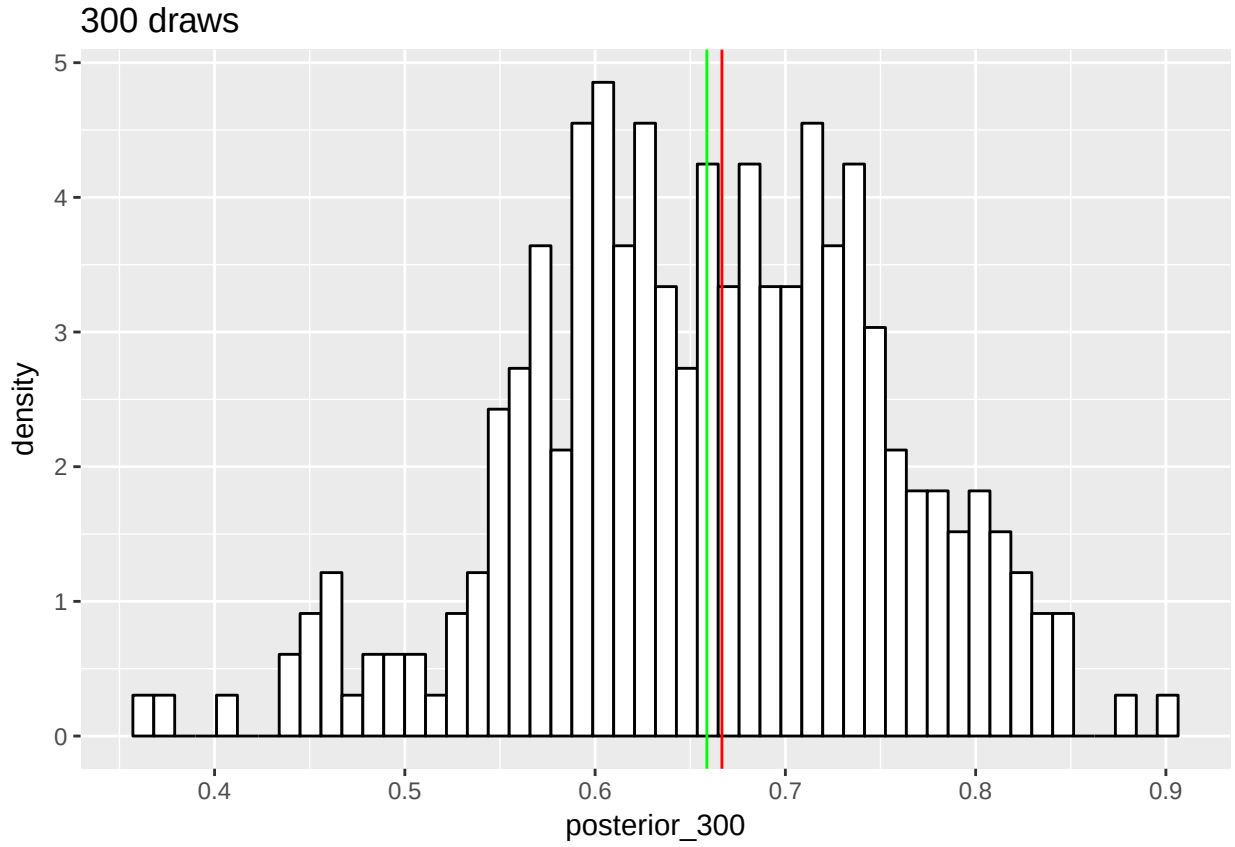
Lab 1 - Bayesian Learning

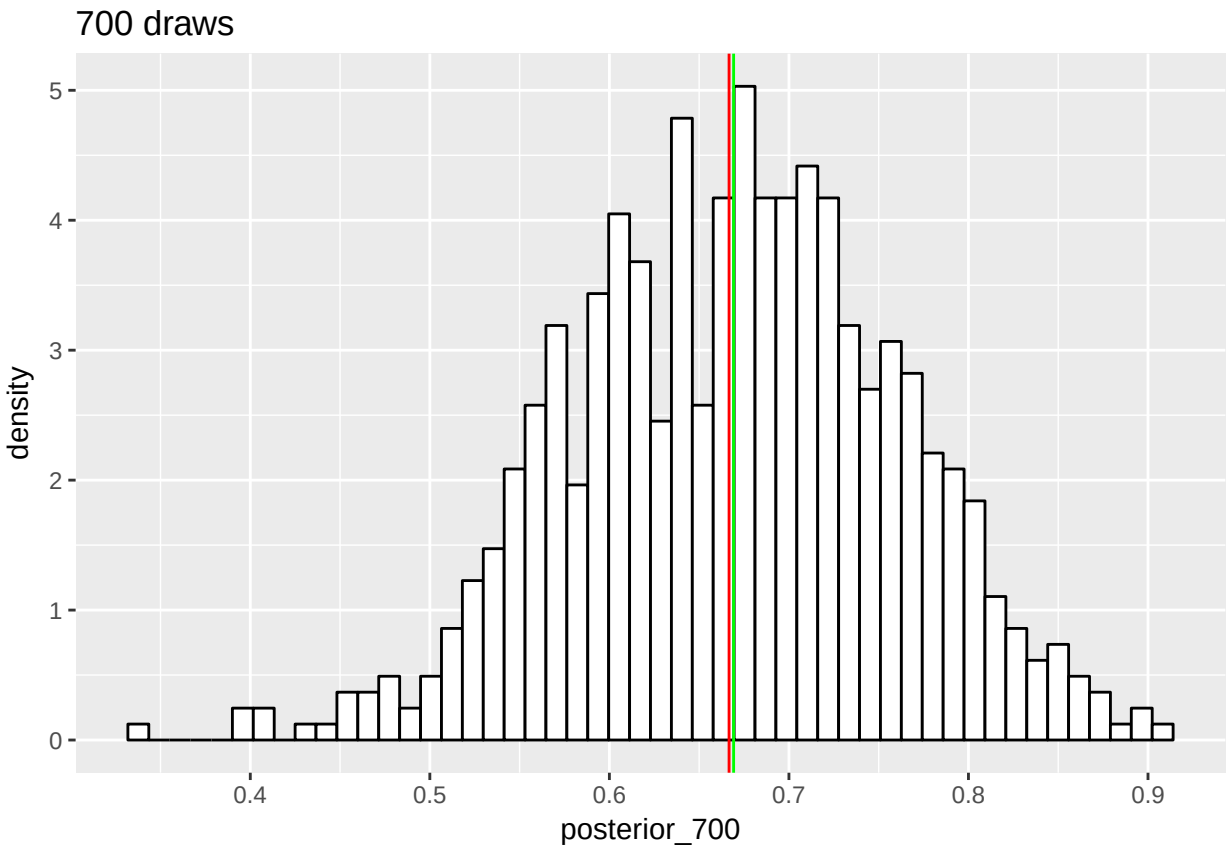
Alessia De Biase and Alejandro Garcia

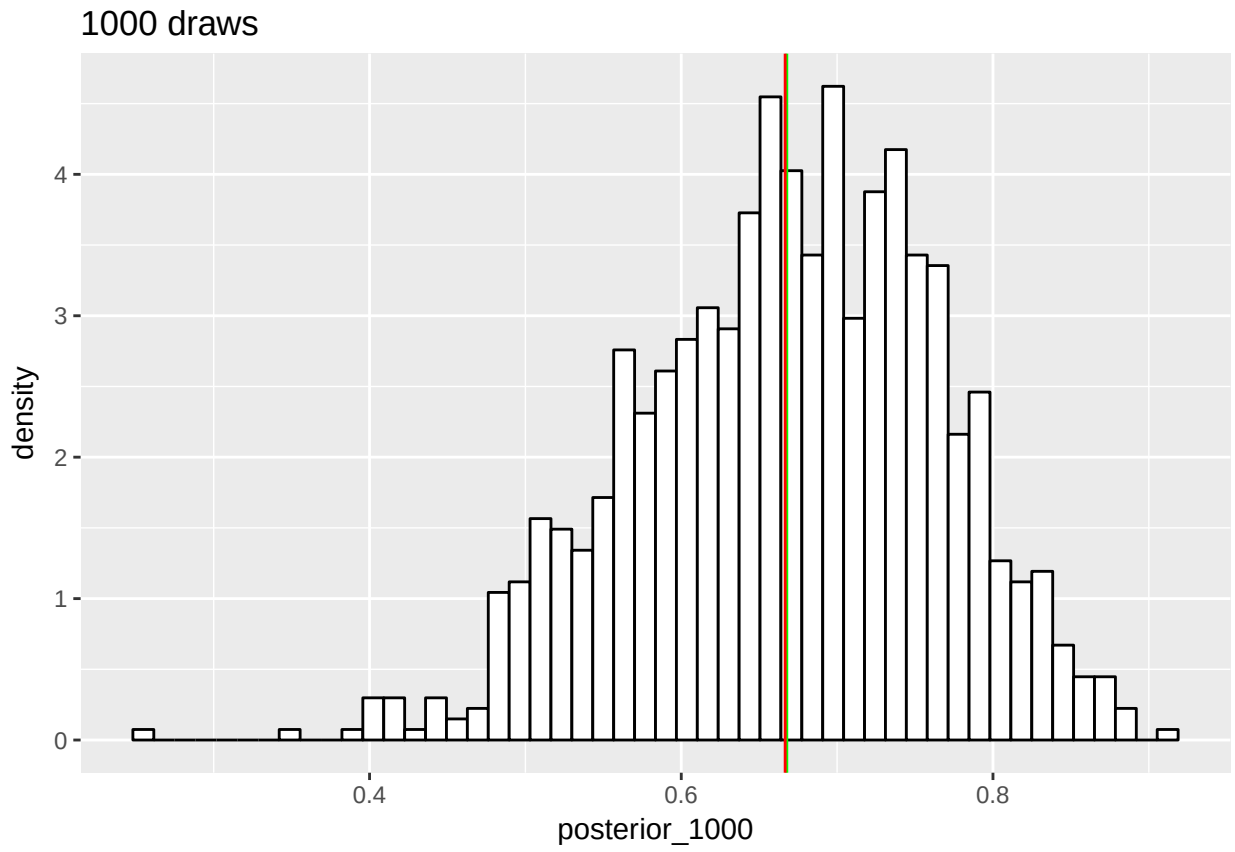
27 marzo 2018

Assignment 1: Bernoulli. . . again

a) In this first step we draw random numbers from the posterior $\theta|y \sim Beta(\alpha_0 + s, \beta_0 + f)$. The following graphs show how increasing the number of random draws the posterior mean (in light green) gets closer and closer to the true mean value (in red).




700 draws
density
posterior_700
0
1
2
3
4
5
0.4
0.5
0.6
0.7
0.8
0.9



Numerically in the following tab we have a better overview of the results from the plots:

Options	Mean	S.Deviation
True Values	0.670	0.090
300 draws	0.659	0.095
700 draws	0.669	0.091
1000 draws	0.668	0.096

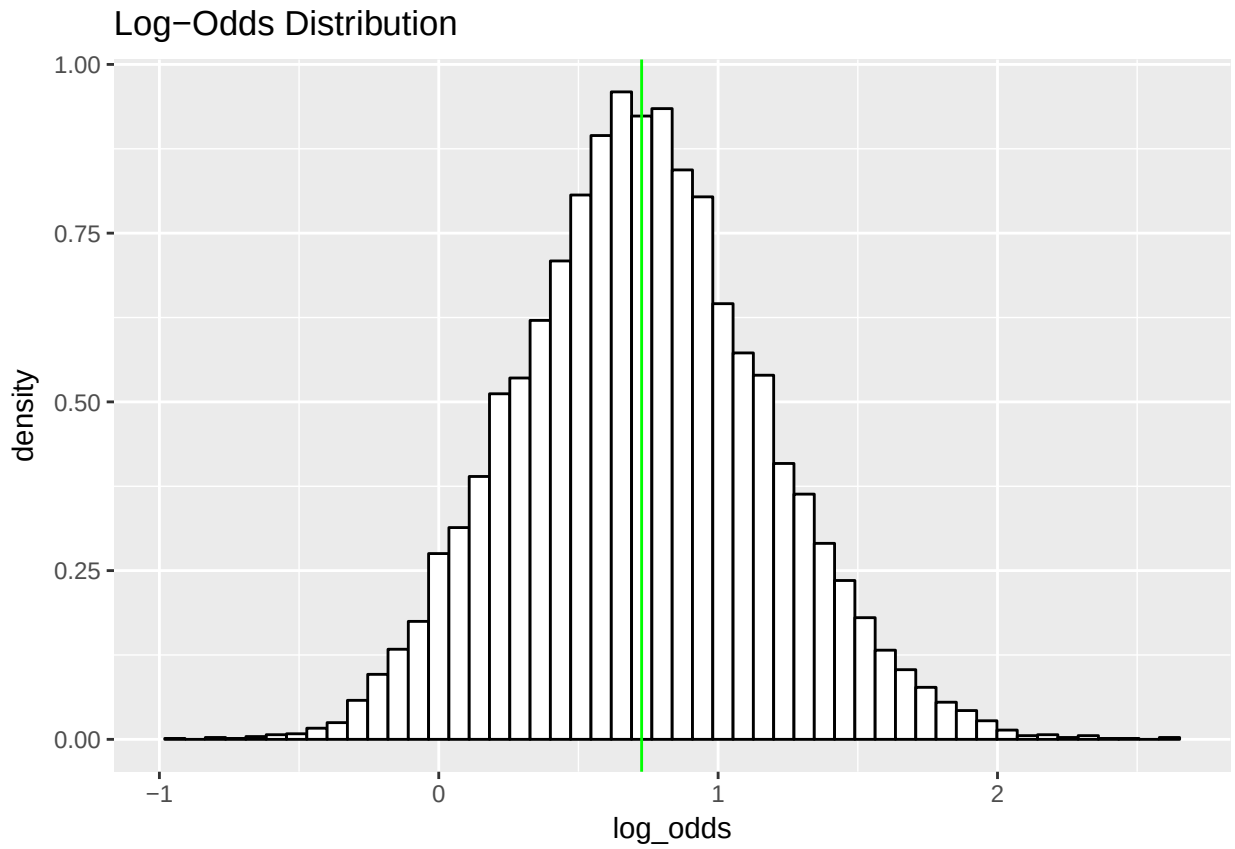
b) We use simulation to compute the posterior probability $P(\theta < 0.4|y)$.

```
nDraws=10000
count=0

for(i in 1:nDraws){
  if(rbeta(1,alpha_0+s,beta_0+n-s)<.4) count=count+1
}

prob_sim=count/nDraws

prob_real=pbeta(.4,alpha_0+s,beta_0+n-s)
```

The posterior probability found is 0.0043 while the exact value is 0.004. As we can see for a number of Draws of 10000 the simulated probability is very closed to the real one.

c) The following graph represents the posterior distribution of the log-odds $\phi = \log(\frac{\theta}{1-\theta})$.

```r
log_odds=c()

for(i in 1:nDraws){
  teta=rbeta(1,alpha_0+s,beta_0+n-s)
  log_odds[i]=log(teta/(1-teta))
}

df_log_odd=data.frame(log_odds=log_odds)

ggplot(df_log_odd,aes(log_odds))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bins
```

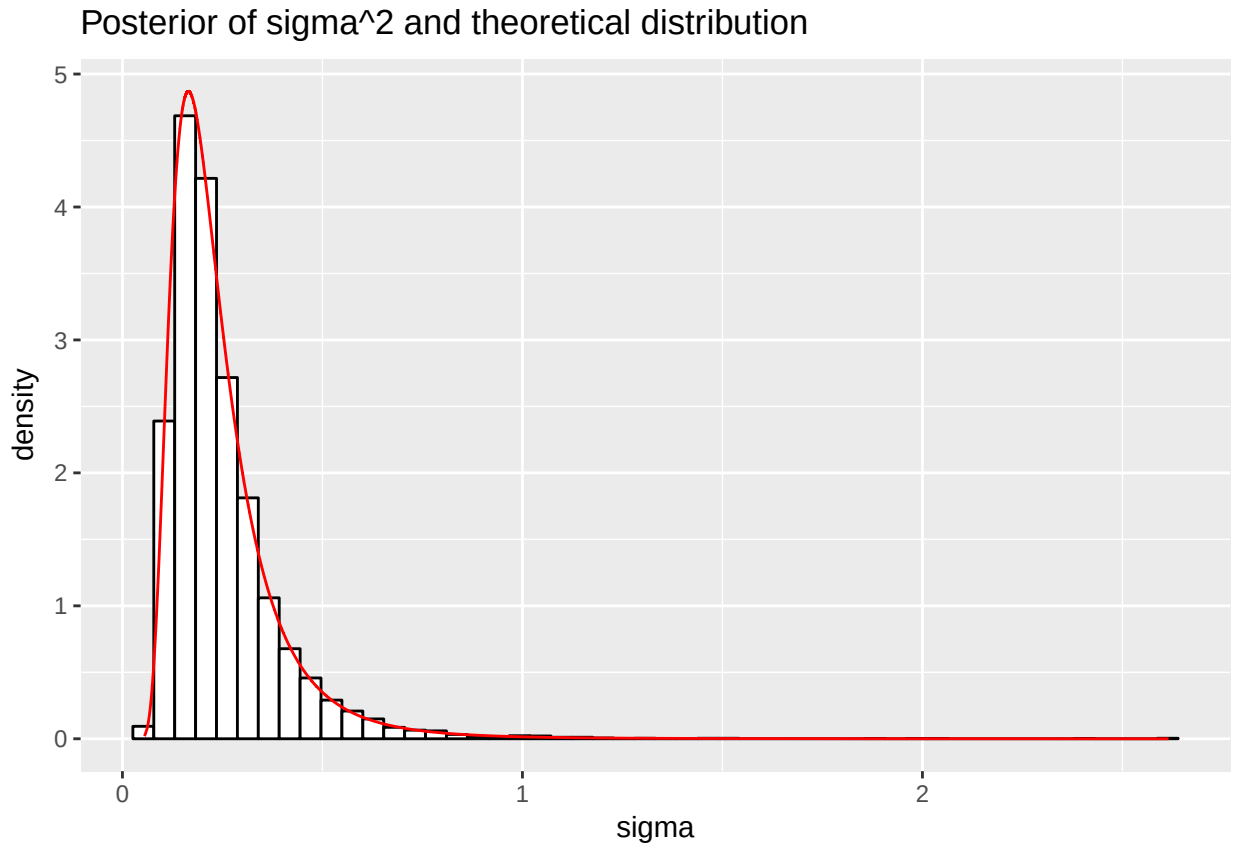


By the graph we can notice how the distribution seams to be normal.

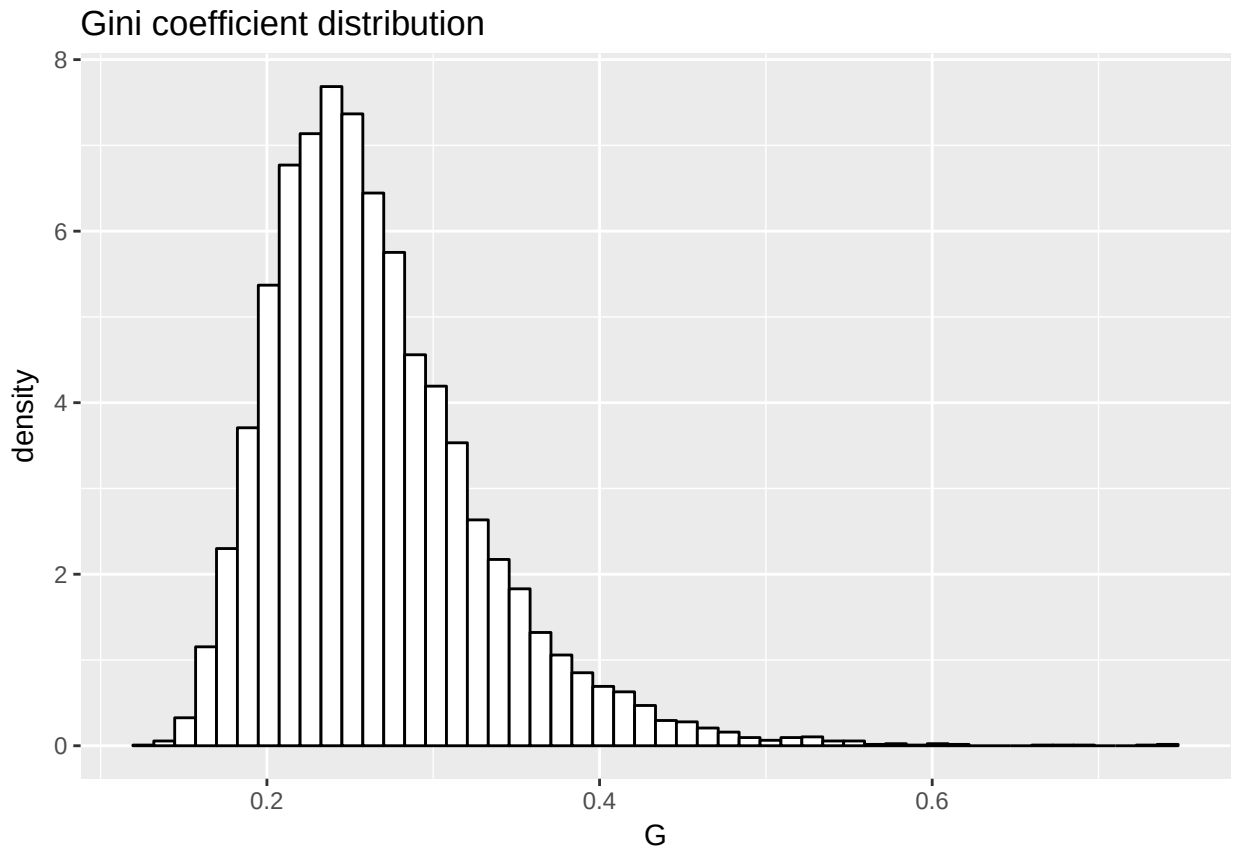
Assignment 2: Log-normal distribution and the Gini coefficient

a) In this first step we simulate 10000 draws from the posterior of σ^2 and we compare the graph with the theoretical posterior distribution. The red line in the graph represents the theoretical posterior distribution, as we can see, it overlaps the histogram perfectly, this is because of the big ammount of data we are using.

```
y=c(14,25,45,25,30,33,19,50,34,67)

mu=3.5
n=length(y)
tau2=sum((log(y)-mu)^2)/n

library(geoR)

post_sigma<-(n*tau2)/rchisq(nDraws,n)
post_mu<-mu+rnorm(nDraws,0,1)*sqrt(post_sigma/n)

PostDraws=data.frame(mu=post_mu,sigma=post_sigma,ver=dinvchisq(post_sigma,df=n,scale=tau2))
```



b) Using the data from the simulations in the previous step, we compute the posterior distribution of the Gini coefficient for the data set. The Gini coefficient measures income inequality, from the graph below we can see that, because all the values are closer to zero than to one, we have a equal income distribution.



c) In this last step we first compute a 95% equal tail credible interval for G and then, using the `density()` function with default settings we use that kernel density estimate to compute a 95% Highest Posterior Density interval for G.

In the first case we obtain the following values: 0.1734531, 0.421202.

In the second case we have 0.1584404, 0.3955984.

The values are very closed to each other.

Assignment 3: Bayesian inference for the concentration parameter in the von Mises distribution.

a) In this assignment we will show that the posterior distribution for some models can be optained by plotting it over a grid of values. The data we have are observed wind directions at a given location on ten different days, they have been converted into radians to fit the description of probability distributions for circular data. The data points are independent observations following the von Mises distribution:

$$p(y|\mu,k) = \frac{\exp[k \cdot \cos(y-\mu)]}{2\pi I_0(k)} \quad -\pi \leq y \leq \pi \tag{1}$$

The likelihood is obtained as it follows:

$$p(y_1,\ldots,y_n|\mu,k) = \prod_{i=1}^{n}[\frac{\exp[k \cdot \cos(y_i-\mu)]}{2\pi I_0(k)}] = \frac{\exp[k \cdot \sum_{i=1}^{n} \cos(y_i-\mu)]}{[2\pi I_0(k)]^n} \tag{2}$$

The prior is $p(k) = \exp(-k)$, because k is Exponential with parameter $\lambda = 1$.

The posterior distribution of k is given by:

$$p(k|y) \propto p(y|\mu,k)p(k) = \frac{\exp[k \cdot \sum_{i=1}^{n} \cos(y_i-\mu)]}{[2\pi I_0(k)]^n} \cdot \exp(-k) \propto \frac{\exp[k \cdot (\sum_{i=1}^{n} \cos(y_i-\mu)-1)]}{[I_0(k)]^n} \tag{3}$$

b) The approximate posterior mode of k corresponds to the highest value of the curve so it is 2.12 (red line in the graph).

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

s=14
n=20
alpha_0=2
beta_0=2

mean=(alpha_0+s)/(alpha_0+beta_0+n)
sd=sqrt(((alpha_0+s)*(beta_0+n-s))/((alpha_0+beta_0+n)^2*(alpha_0+beta_0+n+1)))

df_300=data.frame(posterior_300=rbeta(300,alpha_0+s,beta_0+n-s))
df_700=data.frame(posterior_700=rbeta(700,alpha_0+s,beta_0+n-s))
df_1000=data.frame(posterior_1000=rbeta(1000,alpha_0+s,beta_0+n-s))

library(ggplot2)

ggplot(df_300,aes(posterior_300))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bi

ggplot(df_700,aes(posterior_700))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bi

ggplot(df_1000,aes(posterior_1000))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",
library(kableExtra)
library(knitr)

tab=data.frame(Options=c("True Values","300 draws","700 draws","1000 draws"),Mean=c(round(mean,2),round

kable(tab)

nDraws=10000
count=0

for(i in 1:nDraws){
  if(rbeta(1,alpha_0+s,beta_0+n-s)<.4) count=count+1
}

prob_sim=count/nDraws

prob_real=pbeta(.4,alpha_0+s,beta_0+n-s)


log_odds=c()

for(i in 1:nDraws){
  teta=rbeta(1,alpha_0+s,beta_0+n-s)
  log_odds[i]=log(teta/(1-teta))
}

df_log_odd=data.frame(log_odds=log_odds)

ggplot(df_log_odd,aes(log_odds))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bins
```

```r
y=c(14,25,45,25,30,33,19,50,34,67)

mu=3.5
n=length(y)
tau2=sum((log(y)-mu)^2)/n

library(geoR)

post_sigma<-(n*tau2)/rchisq(nDraws,n)
post_mu<-mu+rnorm(nDraws,0,1)*sqrt(post_sigma/n)

PostDraws=data.frame(mu=post_mu,sigma=post_sigma,ver=dinvchisq(post_sigma,df=n,scale=tau2))


ggplot(PostDraws,aes(sigma))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bins=50)


G=data.frame(G=(2*pnorm(sqrt(post_sigma/2))-1))

ggplot(G,aes(G))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bins=50)+ggtitle("G


#quantile(G$G,probs = c(0.025,0.975)) Equal tail credible interval

#now using density function
dx <- density(G$G)
dn <- cumsum(dx$y)/sum(dx$y)

sorted_prob=sort(dx$y,decreasing = TRUE) #indeces of them in order from highest
ordered_prob=order(dx$y,decreasing=TRUE)

HPD_limit=min(which(cumsum(sorted_prob/sum(dx$y))>=.95))
high_probI=ordered_prob[1:HPD_limit]
HPD=dx$x[c(min(high_probI),max(high_probI))]


x=c(-2.44,2.14,2.54,1.83,2.02,2.33,-2.79,2.23,2.07,2.02)

Mu=2.39

k=data.frame(k=seq(0,10,by=0.01))

posterior=function(k){
  return((exp(k*(sum(cos(x-Mu))-1))/(besselI(k,nu=0))^10))
}

val=as.vector(sapply(k,FUN=posterior))

graph=data.frame(val=val,k=k)

ggplot(data=graph,aes(y=val,x=k))+geom_line(size=1)+ggtitle("Posterior distribution of k")+geom_vline(x
```

Lab 2 - Bayesian Learning

Alessia De Biase and Alejandro Garcia

17 aprile 2018

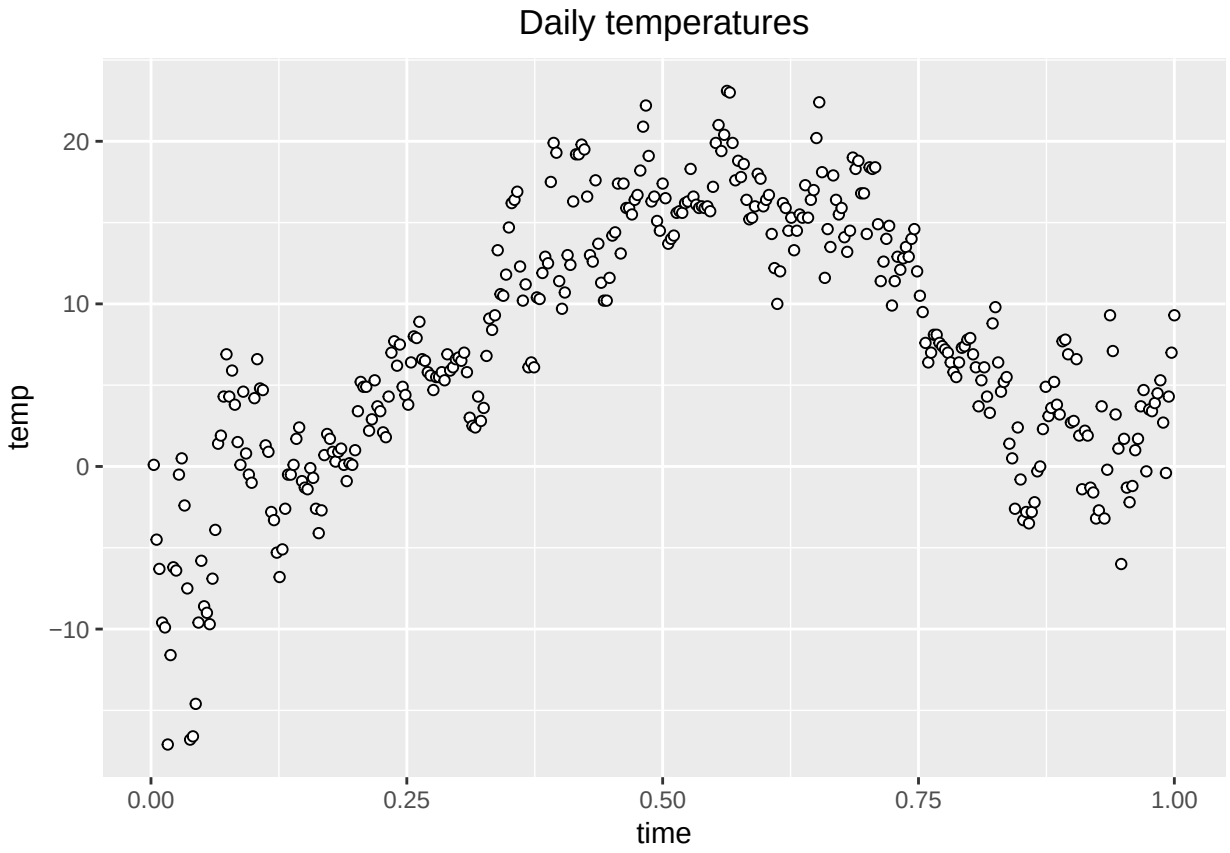
Assignment 1: Linear and polynomial regression

In this first assignment we will perform a Bayesian analysis of a quadratic regression in a dataset containing daily temperatures at Malmslatt, Linkoping, over the course of the year 2016. We will use the dataset `TempLinkoping.txt` which contains two variables: `temp` which is the response variable and `time` which is the covariate.

a) The model we want to perform is the following:

$$temp = \beta_0 + \beta_1 time + \beta_2 time^2 + \epsilon, \quad \epsilon \sim N(0, \sigma^2). \tag{1}$$

As first step we will determine the prior distrbution of the model parameters setting the prior hyperparameters μ_0, Ω_0, ν_0 to sensible values (as suggested we will use the conjugate prior for the linear regression model). Even if by definition the prior shouldn't be biased, we first plot the data to visualize the relation between the two variables and then we set the parameters.



The joint prior for β and σ^2 is:

$$\beta|\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^{-1}) \tag{2}$$

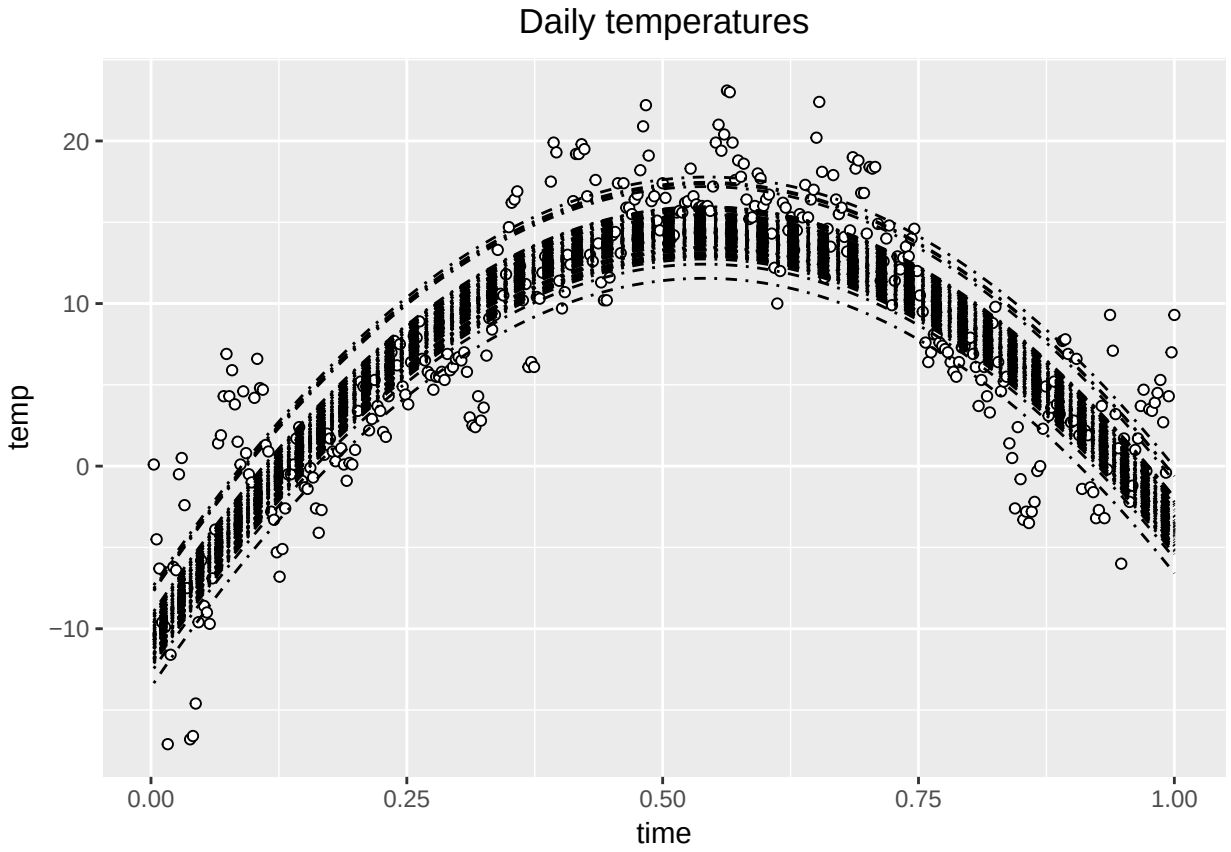
$$\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2) \tag{3}$$

To assign suitable values to the parameters, we first try to interpret how they affect the distribution. After plotting the data we notice that the variance of the β ($\sigma^2\Omega_0^{-1}$) vector should be fixed to low values so also σ^2 should be small (small ν_0 and small σ_0^2). To set the values of the μ_0 vector, instead, we focused on the value of the y-intercept ($\mu_0[1]$), on the angle between the horizontal and model line ($\mu_0[2]$) and on the quadratic term (negative because the curve is concave, $\mu_0[3]$). About Ω_0 , instead, we build a diagonal matrix, each of the value on the main diagonal will represent the denominator of the variance of each of the betas we have.

```
mu0=c(-11,93,-86)
omega0=diag(x=c(1,150,100),nrow=3,ncol=3)
nu0=2.5
sigma0=.5

#lm=lm(temp ~ time + I(time^2),data=temp_Link)
```

b) In this step we check how sensible to the hyperparameters is the prior. We adjusted the initial values to get a more reasonable plot and we obtained the following result:

```
x=temp_Link$time
y=temp_Link$temp
X=matrix(c(rep(1,length(x)),x,x^2),nrow=3,byrow=TRUE)

NDraws=100
for(i in 1:NDraws){
  sigma=(nu0*sigma0)/rchisq(1,nu0)
  #sigma=rinvchisq(1,df=nu0,scale=sigma0)
  beta=rmvnorm(1,mean=mu0,sigma=sigma*solve(omega0))
  model=beta%*%X
  dat=data.frame(model=as.vector(model),x=x)
  q=q+geom_line(aes(x=x,y=model),data=dat,linetype = "dotdash")
}
q
```



c) In this step we write a program which simulates from the joint posterior distribution of $\beta_0, \beta_1, \beta_2$ and σ^2 :

$$\beta|\sigma^2, \mathbf{y} \sim N(\mu_n, \sigma^2\Omega_n^{-1}) \tag{4}$$

$$\sigma^2|\mathbf{y} \sim Inv - \chi^2(\nu_n, \sigma_n^2) \tag{5}$$

where:

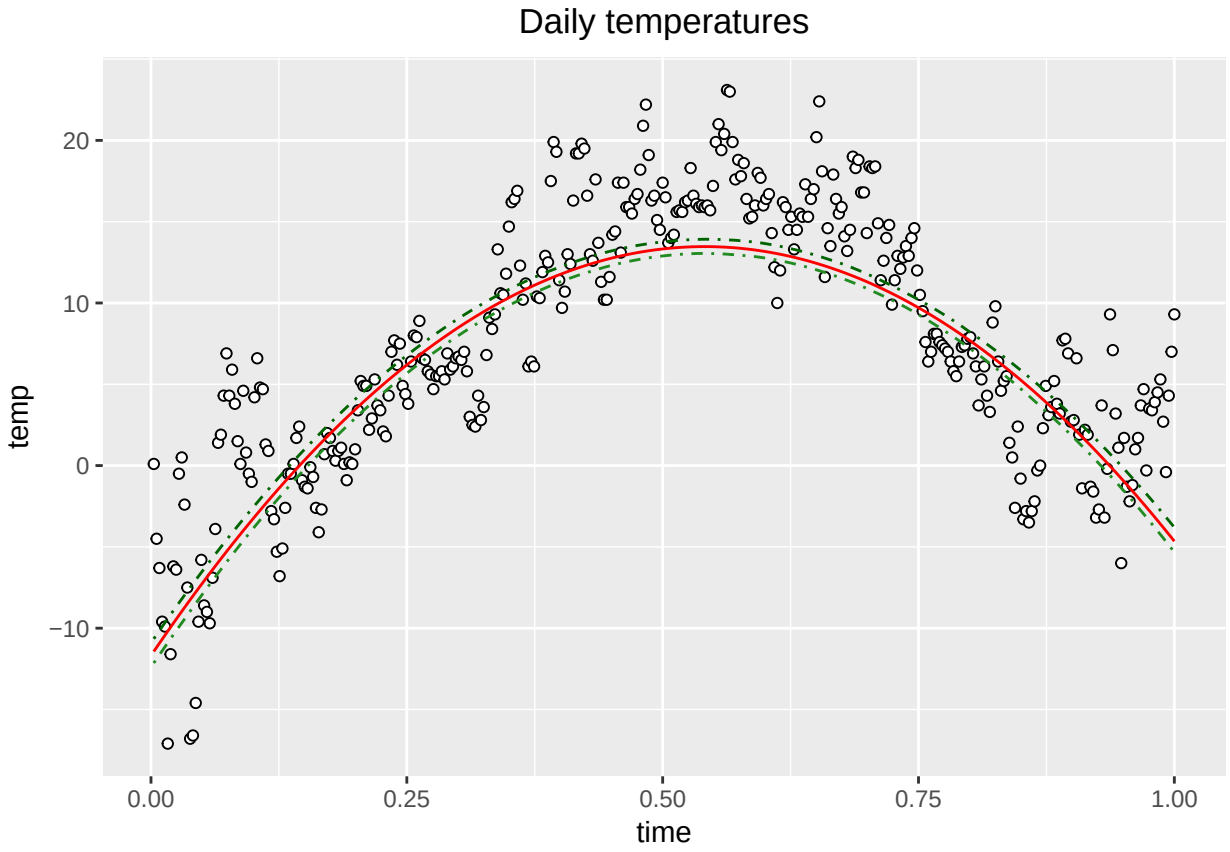
$$\mu_n = (\mathbf{X}'\mathbf{X} + \Omega_0)^{-1}(\mathbf{X}'\mathbf{X}\hat{\beta} + \Omega_0\mu_0), \quad \Omega_n = \mathbf{X}'\mathbf{X} + \Omega_0 \tag{6}$$

$$\nu_n = \nu_0 + n, \quad \nu_n\sigma_n^2 = \nu_0\sigma_0^2 + (\mathbf{y}'\mathbf{y} + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n) \tag{7}$$

In the following plot the red line represents the posterior mean of the quadratic regression function; the upper green dotted line is the upper 95% posterior credible interval for the regression function while the lower corresponds to the lower 5% posterior credible interval. As we can see in between those two last lines there are not all the observations because the original plot refers to the data but the confidence intervals are built on the posterior simulations.

```
model_post=matrix(nrow=NDraws,ncol=n)
beta_post=matrix(nrow=NDraws,ncol=3)

#simulating from joint posterior distribution
for(i in 1:NDraws){
```

```
  sigma_post=(nu_n*sigma_n)/rchisq(1,nu_n)
  beta_post[i,]=rmvnorm(1,mean=mu_n,sigma=sigma_post*solve(omega_n))
  model_post[i,]=beta_post[i,]%*%X #predicted posterior temperatures
}

avg_post=as.vector(t(colMeans(model_post)))
post_AVG=data.frame(avg_post=avg_post,x=x)

#low 5% and up 95%
first_ci=apply(model_post,2,quantile,c(0.05,.95))
```



d) The model is a quadratic curve then the value of time corresponding to the maximum value can be found as $x = \frac{-\beta_1}{2\beta_2}$. From the previous simulations we obtain that on average the highest temperature is obtained when x is equal to:

```
x_tilda=-beta_post[,2]/(2*beta_post[,3])

mean(x_tilda)
```

```
## [1] 0.5409285
```

This means that the highest temperature found in the simulations from the posterior happens around July.

e) If we want to estimate a polynomial model of order 7 it would look like:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_7 x^7 \tag{8}$$

. The μ_0 vector will have the first three values equal to the ones in step 1 and zero from the fourth to

the seventh position; the Ω_0 matrix will still be diagonal but it will have very high values corresponding to all the betas after β_2 on the diagonal (high values on the diagonal of Ω correspond to low values for the variance of the parameters - shrinkage effect).

Assignment 2: Posterior approximation for classification with logistic regression

In this second assignment we use the dataset `WomenWork.dat` which contains 200 observations on nine variables about women. The main aim is to predict if a woman with certain characteristics is more likely to work or not.

a) The model used in this assignment is the logistic regression because the response variable is a binary variable (1 if the woman works, 0 if she doesn't):

$$Pr(y = 1|x) = \frac{\exp(x^T\beta)}{1 + \exp(x^T\beta)} \tag{9}$$

We first fit the logistic regression using maximum likelihood estimation.

```
WomenWork=read.table("WomenWork.txt",header=TRUE)

y=as.vector(WomenWork$Work)
X=as.matrix(WomenWork[,-1])

glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial)

summary(glmModel)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = WomenWork)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1662  -0.9299   0.4391   0.9494   2.0582
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.64430    1.52307   0.423 0.672274
## HusbandInc  -0.01977    0.01590  -1.243 0.213752
## EducYears    0.17988    0.07914   2.273 0.023024 *
## ExpYears     0.16751    0.06600   2.538 0.011144 *
## ExpYears2   -0.14436    0.23585  -0.612 0.540489
## Age         -0.08234    0.02699  -3.050 0.002285 **
## NSmallChild -1.36250    0.38996  -3.494 0.000476 ***
## NBigChild   -0.02543    0.14172  -0.179 0.857592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 200  degrees of freedom
## Residual deviance: 222.73  on 192  degrees of freedom
## AIC: 238.73
##
## Number of Fisher Scoring iterations: 4
```

From the output we notice already which are the significative variables (`NSmallChild`,`Age` and `EducYears`,`ExpYears`).

b) In this second step we want to approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution:

$$\beta|\mathbf{y},\mathbf{X} \sim N(\tilde{\beta}, J_y^{-1}(\tilde{\beta})) \tag{10}$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta})$ is the observed Hessian evaluated at the posterior mode.

The prior we will use in this case is $\beta \sim N(0, \tau^2 I)$ with $\tau = 10$.

The likelihood function, instead, is:

$$p(Y_i = y|X_i,\beta) = (\frac{e^{\beta X_i}}{1+e^{\beta X_i}})^y (1 - \frac{e^{\beta X_i}}{1+e^{\beta X_i}})^{1-y} = \frac{e^{\beta X_i y}}{1+e^{\beta X_i}} \tag{11}$$

$$L(\beta|x) = p(\mathbf{Y}|\mathbf{X};\beta) = \prod_i Pe(y_i|x_i,\beta) = \prod_i \frac{e^{\beta X_i y}}{1+e^{\beta X_i}} \tag{12}$$

The logLikelihood function is:

$$logL(\beta|x) = \log(\prod_i \frac{e^{\beta X_i y}}{1+e^{\beta X_i}}) = \sum_i (\beta X_i y - \log(1+\exp(\beta X_i))) \tag{13}$$

The posterior we want to maximize is:

$$p(\beta|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X};\beta)p(\beta) \tag{14}$$

The logarithm of the posterior will be then proportional to the sum of the logarithms of the prior and the likelihood so in our code we will use the function `LogPostLogistic` which will return the sum of them.

```
# Prior:
nPara=dim(X)[2]

#Parameters
tau=10
mu <- as.vector(rep(0,nPara))
Sigma <- tau^2*diag(nPara)

#Post LogLikelihood
LogPostLogistic <- function(betaVect,y,X,mu,Sigma){
  dim <- length(betaVect);
  linPred <- X%*%betaVect;
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000;
  logPrior <- dmvnorm(betaVect, matrix(0,dim,1), Sigma, log=TRUE);
  return(logLik + logPrior)
}

start <- as.vector(rep(0,nPara));

Optim<-optim(start,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),
             control=list(fnscale=-1),hessian=TRUE)

approxPostStd <- sqrt(diag(-solve(Optim$hessian))) # Computing approximate standard deviations.
```

Using the function `optim` we obtain that the posterior mode is 0.63, -0.02, 0.18, 0.17, -0.14, -0.08, -1.36, -0.02 and the approximate posterior standard deviation is 1.51, 0.02, 0.08, 0.07, 0.24, 0.03, 0.39, 0.14 .

For the variable `NSmallChild`, the most significant variable as we noticed from a), we now compute an approximate 95% credible interval:

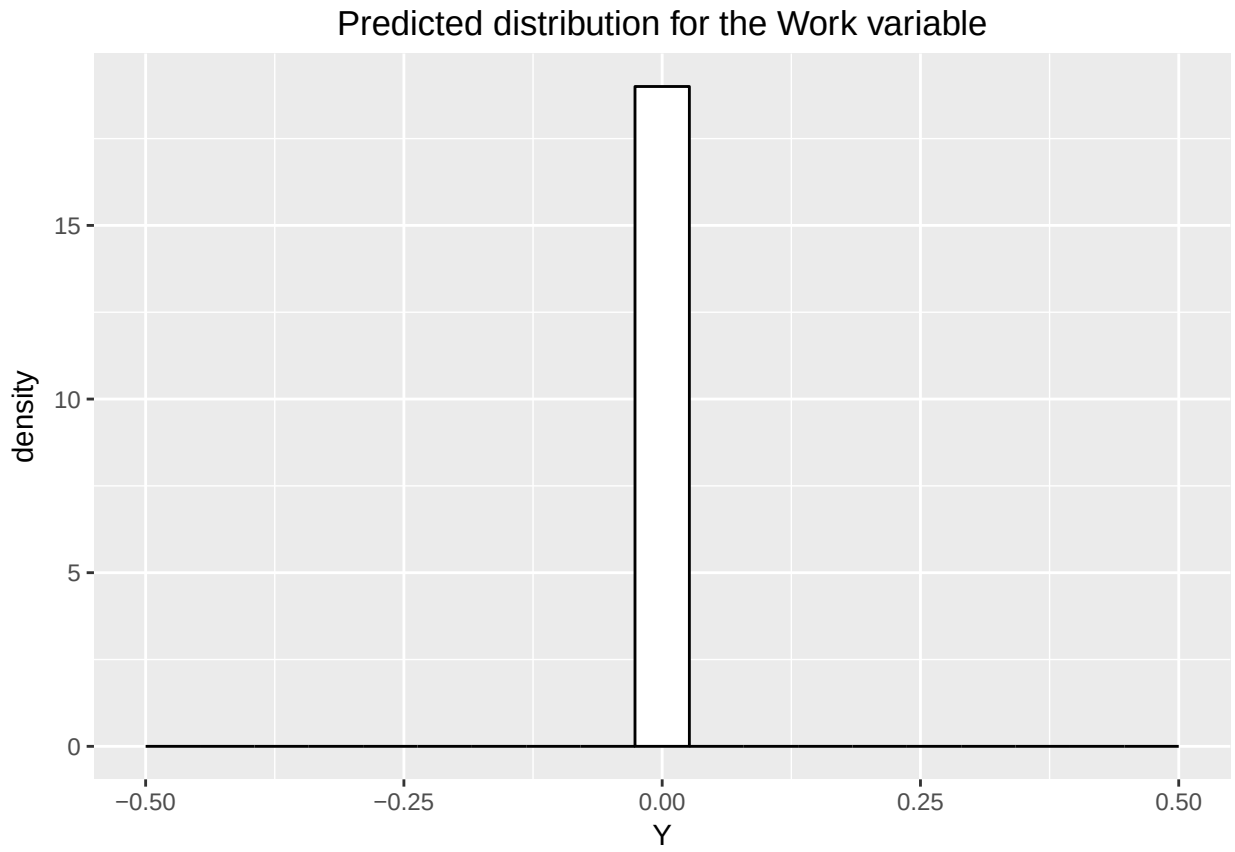
```
## [1] -2.1214250 -0.5968414
```

This feature is the one which affects the most the response variable.

c) In this last step we write a function `predicted_logReg` which simulates from the predictive distribution of the response variable in a logistic regression. The function takes the vector `x` with the new observation, `mu` and `J` which are the parameters for the posterior normal distribution. After generating a draw from the posterior we generate the probability of $y = 1$ and this will determine if the variable work should be 1 or 0.

```
#Woman characteristics
x=as.matrix(c(1,10,8,10,1,40,1,1))

predicted_logReg=function(x,mu,J){

  set.seed(12345)
  post=rmvnorm(1,mean=mu,sigma=J)
  p=exp(t(x)%*%t(post))/(1+exp(t(x)%*%t(post)))
  return((p>=runif(1))*1)

}

NDraws=10000
Y=c()
for(i in 1:NDraws){
  set.seed(12345)
  Y[i]=predicted_logReg(x,mu=Optim$par,J=-solve(Optim$hessian))
}
```

The following plot represents the predictive distribution for the `Work` variable for a 40 year old woman with two children (3 and 9 years old), 8 years of education, 10 years of experience and an husband with an income of 10.



From the graph we notice that the distribution of the `Work` variable is a Bernoulli with parameter p around 0. The probability that a woman with this characteristics wouldn't work is pretty high.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

library(ggplot2)
library(mvtnorm)
library(geoR)

set.seed(12345)

temp_Link=read.csv(file="TempLinkoping.csv",sep=";")

p=ggplot()+geom_point(data=temp_Link,aes(x=time,y=temp),colour = "black", fill = "white",shape=21)+
  labs(title="Daily temperatures")+theme(plot.title = element_text(hjust = 0.5))
q=p
p


mu0=c(-11,93,-86)
omega0=diag(x=c(1,150,100),nrow=3,ncol=3)
nu0=2.5
sigma0=.5

#lm=lm(temp ~ time + I(time^2),data=temp_Link)
x=temp_Link$time
y=temp_Link$temp
X=matrix(c(rep(1,length(x)),x,x^2),nrow=3,byrow=TRUE)

NDraws=100
for(i in 1:NDraws){
  sigma=(nu0*sigma0)/rchisq(1,nu0)
  #sigma=rinvchisq(1,df=nu0,scale=sigma0)
  beta=rmvnorm(1,mean=mu0,sigma=sigma*solve(omega0))
  model=beta%*%X
  dat=data.frame(model=as.vector(model),x=x)
  q=q+geom_line(aes(x=x,y=model),data=dat,linetype = "dotdash")
}
q
n=length(x)
mu_n=solve(X%*%t(X)+omega0)%*%(X%*%(t(X)%*%t(beta))+omega0%*%mu0)
omega_n=X%*%t(X)+omega0
nu_n=nu0+n
sigma_n=as.numeric((nu0*sigma0+(t(y)%*%y)+t(mu0)%*%omega0%*%mu0-t(mu_n)%*%omega_n%*%mu_n)/nu_n)

model_post=matrix(nrow=NDraws,ncol=n)
beta_post=matrix(nrow=NDraws,ncol=3)

#simulating from joint posterior distribution
for(i in 1:NDraws){
  sigma_post=(nu_n*sigma_n)/rchisq(1,nu_n)
  beta_post[i,]=rmvnorm(1,mean=mu_n,sigma=sigma_post*solve(omega_n))
  model_post[i,]=beta_post[i,]%*%X #predicted posterior temperatures
}
```

```r
avg_post=as.vector(t(colMeans(model_post)))
post_AVG=data.frame(avg_post=avg_post,x=x)

#low 5% and up 95%
first_ci=apply(model_post,2,quantile,c(0.05,.95))


CI <- data.frame(first=first_ci[1,],second=first_ci[2,],x=x)

p=p+geom_line(data=post_AVG,aes(x=x,y=avg_post),col="red")+geom_line(data=CI,aes(x=x,y=first),col="fores
  geom_line(data=CI,aes(x=x,y=second),col="dark green",linetype = "dotdash")

p
x_tilda=-beta_post[,2]/(2*beta_post[,3])

mean(x_tilda)

WomenWork=read.table("WomenWork.txt",header=TRUE)

y=as.vector(WomenWork$Work)
X=as.matrix(WomenWork[,-1])

glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial)

summary(glmModel)

# Prior:
nPara=dim(X)[2]

#Parameters
tau=10
mu <- as.vector(rep(0,nPara))
Sigma <- tau^2*diag(nPara)

#Post LogLikelihood
LogPostLogistic <- function(betaVect,y,X,mu,Sigma){
  dim <- length(betaVect);
  linPred <- X%*%betaVect;
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000;
  logPrior <- dmvnorm(betaVect, matrix(0,dim,1), Sigma, log=TRUE);
  return(logLik + logPrior)
}

start <- as.vector(rep(0,nPara));

Optim<-optim(start,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),
             control=list(fnscale=-1),hessian=TRUE)

approxPostStd <- sqrt(diag(-solve(Optim$hessian))) # Computing approximate standard deviations.

Exp_NSmallChild=Optim$par[7]
Var_NSmallChild=approxPostStd[7]
```

```r
PCI_NSmallChild_U=Exp_NSmallChild+1.96*Var_NSmallChild #Upper
PCI_NSmallChild_D=Exp_NSmallChild-1.96*Var_NSmallChild #Lower

print(c(PCI_NSmallChild_D,PCI_NSmallChild_U))

#Woman characteristics
x=as.matrix(c(1,10,8,10,1,40,1,1))

predicted_logReg=function(x,mu,J){

  set.seed(12345)
  post=rmvnorm(1,mean=mu,sigma=J)
  p=exp(t(x)%*%t(post))/(1+exp(t(x)%*%t(post)))
  return((p>=runif(1))*1)

}

NDraws=10000
Y=c()
for(i in 1:NDraws){
  set.seed(12345)
  Y[i]=predicted_logReg(x,mu=Optim$par,J=-solve(Optim$hessian))
}

df=data.frame(y=Y)

ggplot(df,aes(Y))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",bins=20)+
  labs(title="Predicted distribution for the Work variable")+theme(plot.title = element_text(hjust = 0.5
```

Lab 3 - Bayesian Learning

Alessia De Biase and Alejandro Garcia

17 aprile 2018

Assignment 1: Normal model, mixture of normal model with semi-conjugate prior

In this assignment we will use the data `rainfall.dat` which consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation at Snoqualmie Falls in Washington. We will examine those data using two models: normal and mixture normal model.

a) *Normal model*

The daily precipitation $y_1, \ldots, y_n$ are independent normally distributed, $y_1, \ldots, y_n | \mu, \sigma^2 \sim N(\mu, \sigma^2)$ where μ and σ^2 are unknown. Let $\mu \sim N(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$.

In the following code we will implement a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2 | y_1, \ldots, y_n)$. The conditional posteriors are the following:

$$\mu | \sigma^2, x \sim N(\mu_n, \tau_n^2) \tag{1}$$

$$\sigma^2 | \mu, x \sim Inv - \chi^2(\nu_n, \frac{\nu_0 \sigma_0^2 + \sum_{i=1}^{n} (x_i - \mu)^2}{n + \nu_0}) \tag{2}$$

μ_n and τ_n^2 are defined as:

$$\frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2} \tag{3}$$

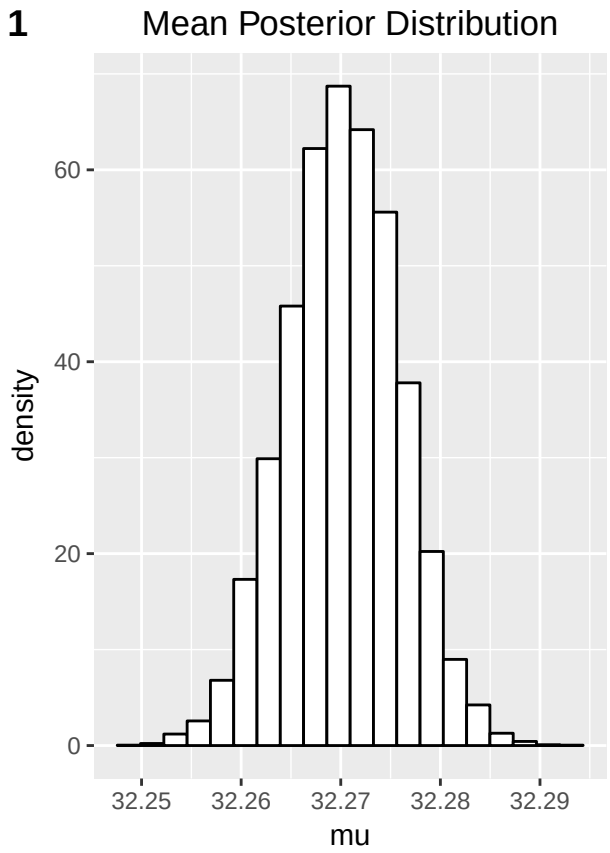
$$\mu_n = w\bar{x} + (1 - w)\mu_0 \tag{4}$$

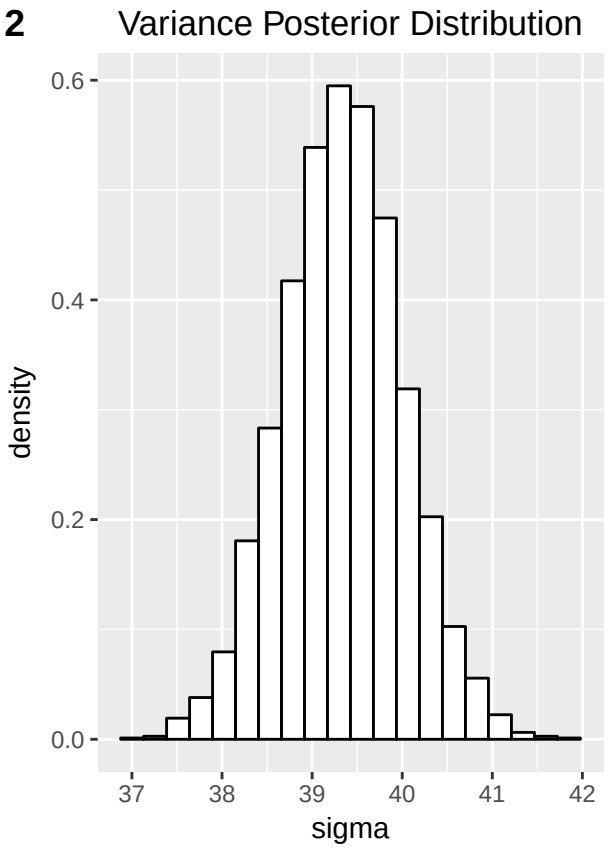
$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{n}{\tau_0^2}} \tag{5}$$

```
set.seed(12345)

n=length(x)

#initialization of parameters
mu0=30 #plotting the data the mean value looks closed to around 30
tau0=1
nu0=1
sigma0=1
NIt=10000 #n° iterations

#first value of sigma given
sigma=rinvchisq(1,df=nu0,scale=sigma0)
```

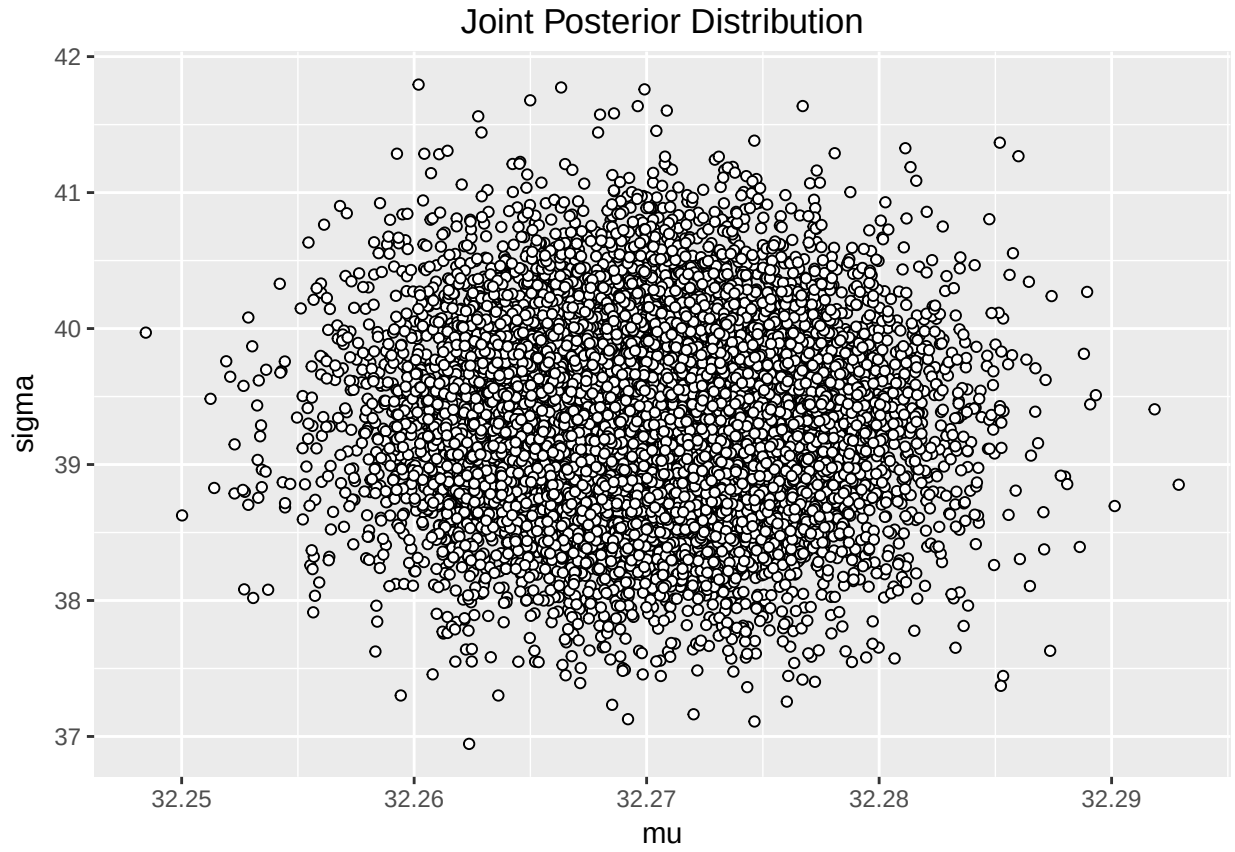
```
join=matrix(ncol=2,nrow = NIt)

for(i in 1:NIt){

  w=(n/sigma)/((n/sigma)+(1/tau0))
  mu_n=w*mean(x)+(1-w)*mu0
  tau_n=1/((n/sigma)+(1/tau0)) #it's squared
  nu_n=nu0+n

  mu=rnorm(1,mean=mu_n,sd=tau_n)

  scal=((nu0*sigma0)+sum((x-mu)^2))/(nu_n)
  sigma=rinvchisq(1,df=nu_n,scale=sqrt(scal))

  join[i,1]=mu
  join[i,2]=sigma

}
```

Now using the Gibbs sampler from the previous step we will first look at the mean distribution and the variance distribution, both look normal.

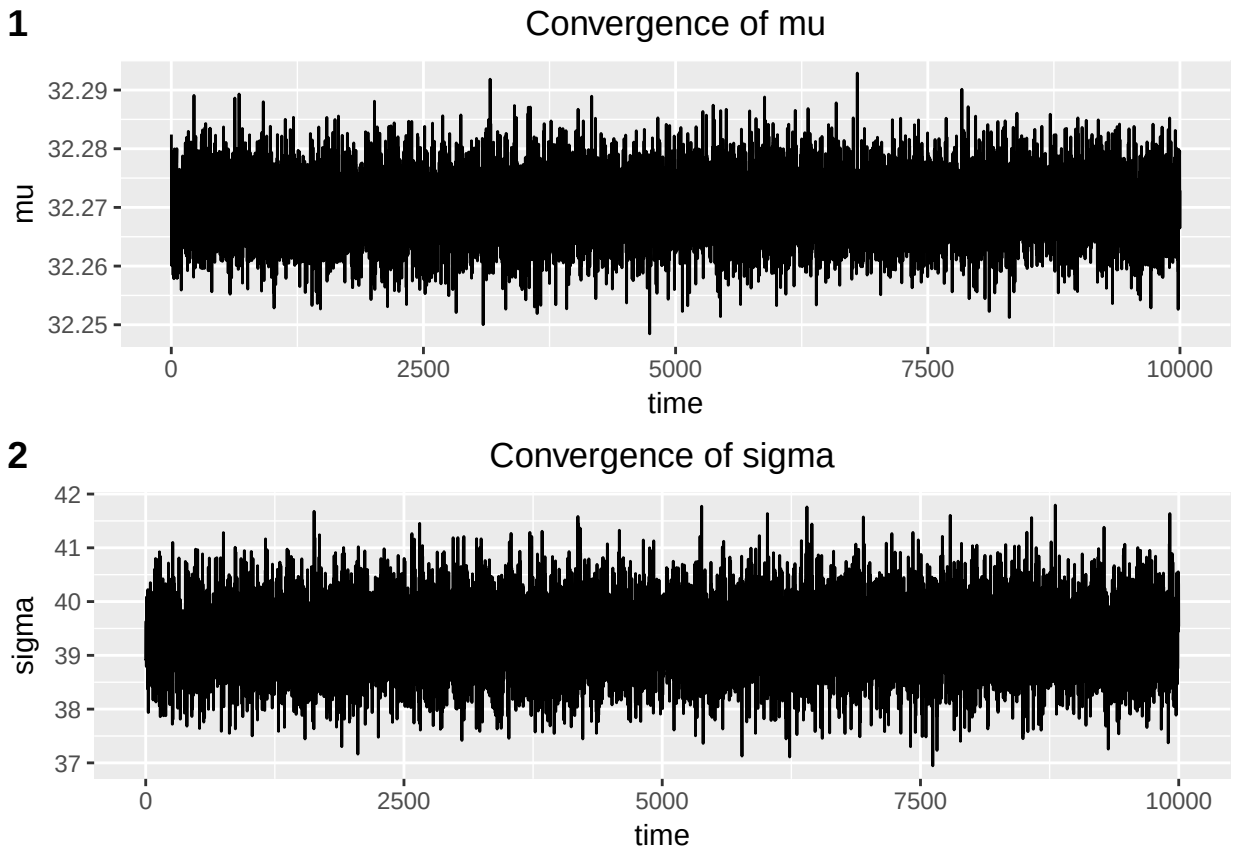




Now we represent the joint posterior distribution, the oval shape suggests a normal distribution.



And in this last step we look at the convergence of the Gibbs sampler:



Both sigma and mu converge pretty fast, this is because the burning period is very small so the prior is closed to the actual distribution of data.

b) *Mixture normal model*

The daily precipitation $y_1, \ldots, y_n$ follow an iid two-component mixture of normals model:

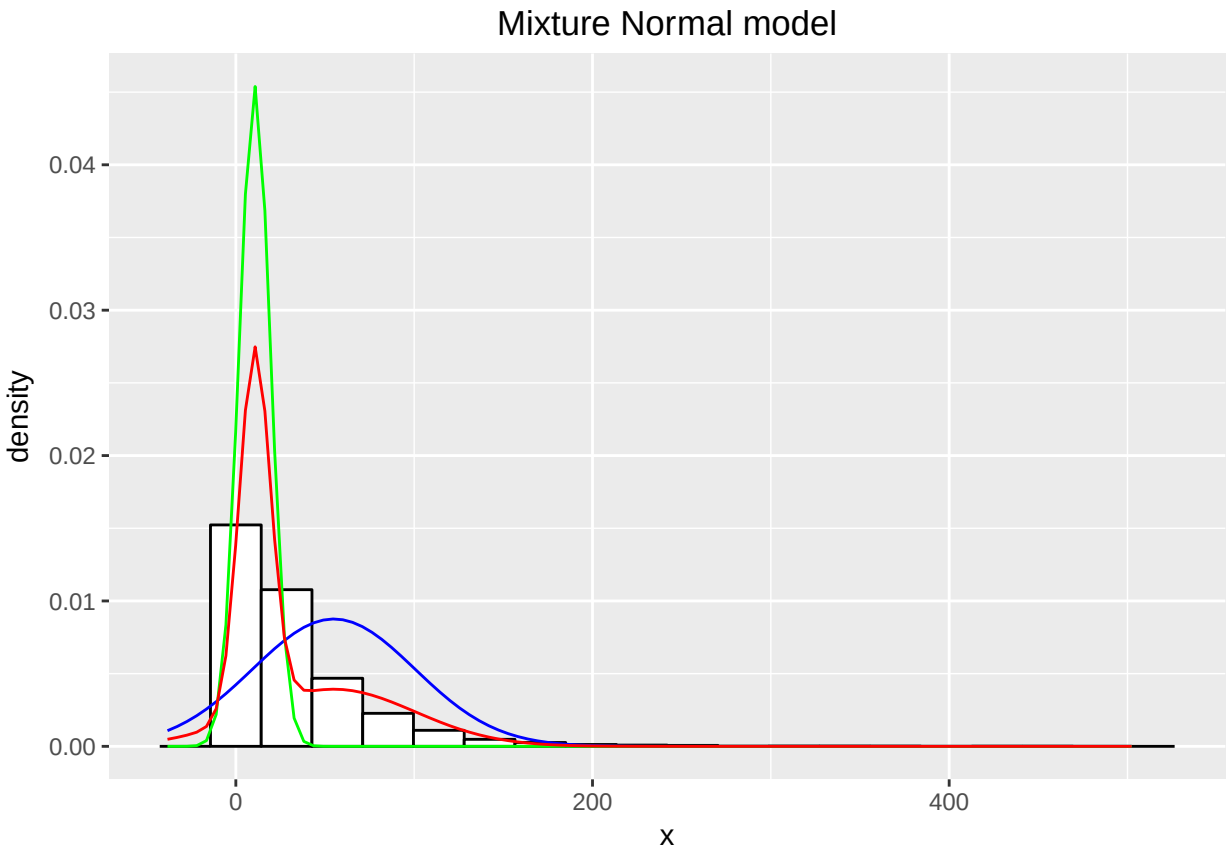
$$p(y_i|\mu, \sigma^2, \pi) = \pi N(y_i|\mu_1, \sigma_1^2) + (1 - \pi)N(y_i|\mu_2, \sigma_2^2) \tag{6}$$

where

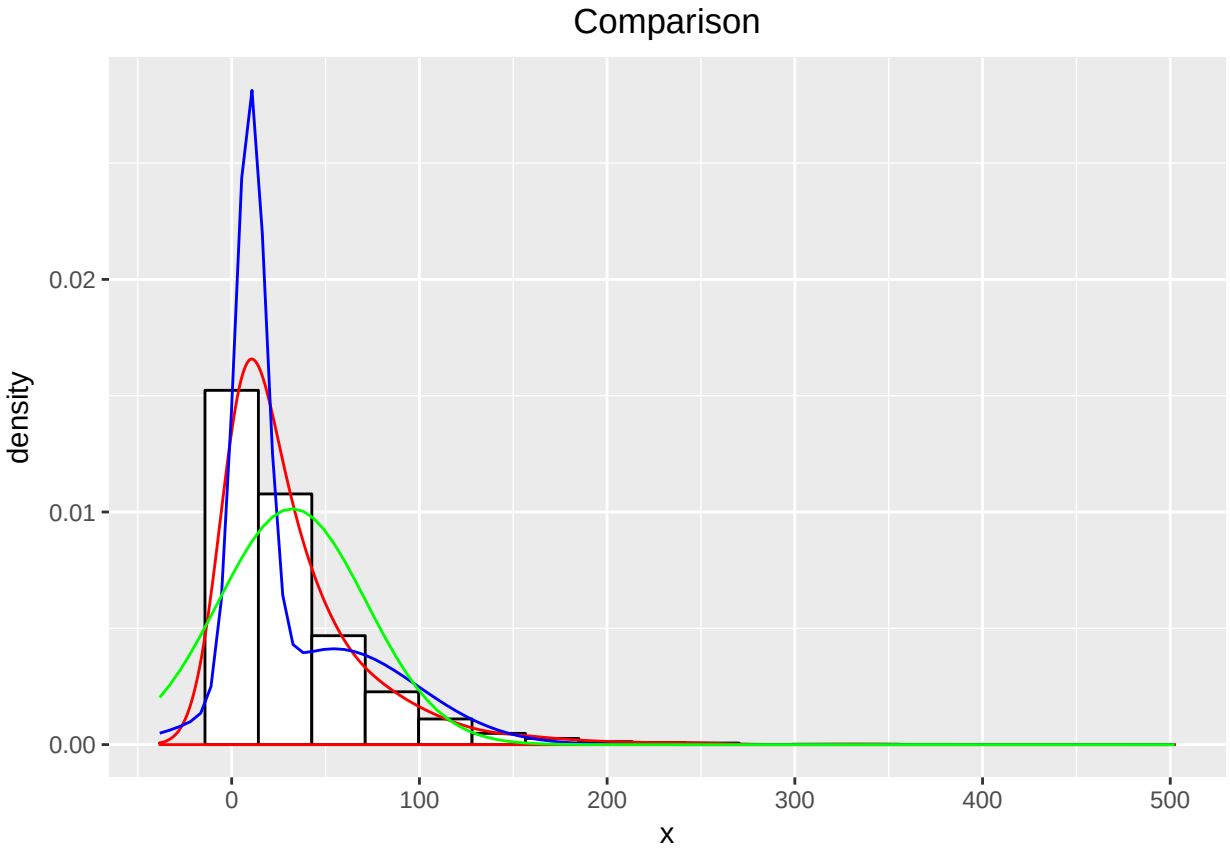
$$\mu = (\mu_1, \mu_2) \quad and \quad \sigma^2 = (\sigma_1^2, \sigma_2^2) \tag{7}$$

We now use the Gibbs sampling data augmentation algorithm to analyze the daily precipitation data.

The blue and the green lines represent the two normal component, the red line represent the mixture of normals.



c) This last plot contains an histogram or kernel estimate of the data (in red), the normal density from a) (in green) and the mixture of normals from b) (in blue). Visually the blue lines seams to fit better the distribution.


Comparison
density
0.02
0.01
0.00
0
100
200
300
400
500
x

Assignment 2: Time series models in Stan.

In this assignment we will work with time series. The process we will use to generate data is an *AR(1)* process of the following form:

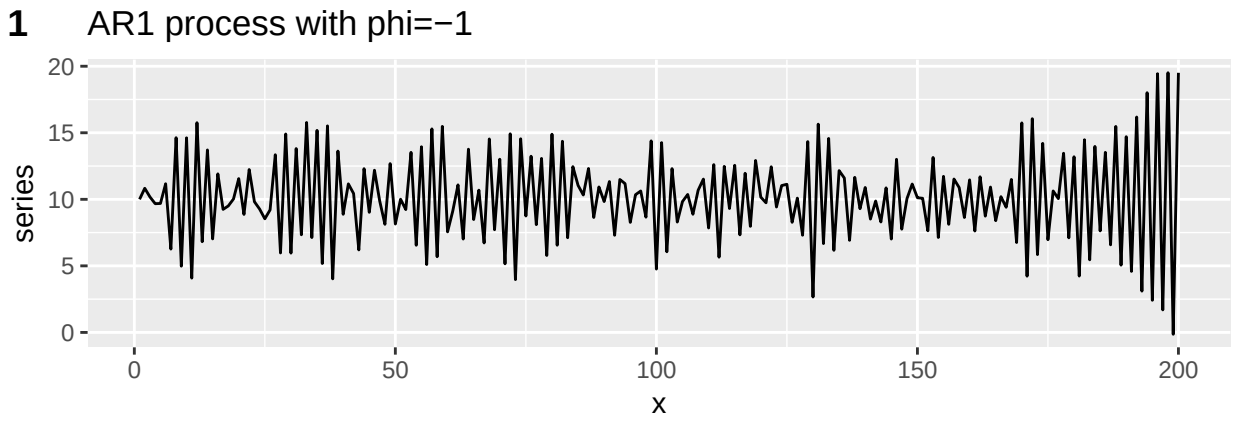
$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2) \tag{8}$$

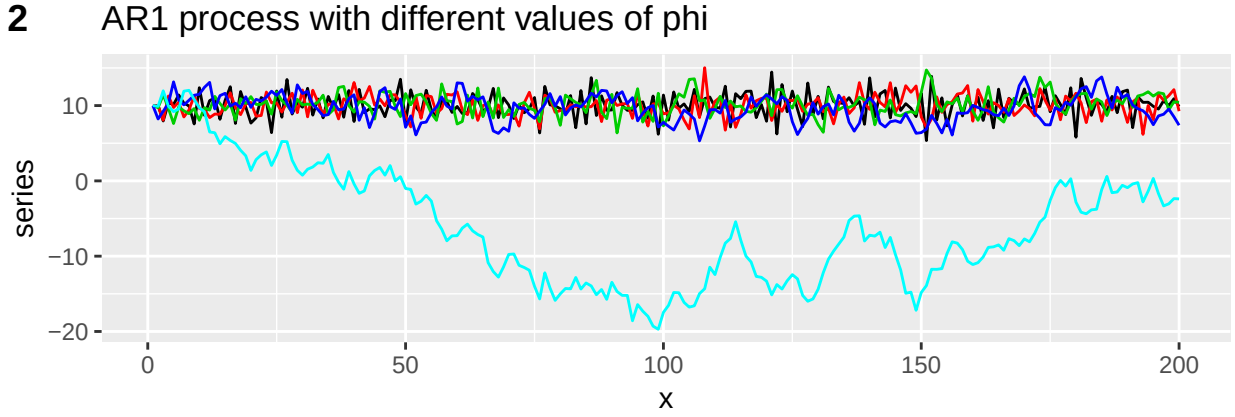
for given values of μ, ϕ, σ^2.

a) In this first step we write a function which simulates values for x_t for $t = 2, 3, \ldots, T$ and which returns the vector $x_{1:T}$ with all data points. Our starting point will be $x_1 = \mu$.

```
set.seed(12345)

AR1=function(mu,phi,sigma,T){
  x<-vector()
  x[1]=mu
  for(i in 2:T){
    eps=rnorm(1,mean=0,sd=sqrt(sigma))
    x[i]=mu+phi*(x[i-1]-mu)+eps
  }
  return(data.frame(x=seq(1:length(x)),series=as.vector(x)))
}
```

We fix $\mu = 10, \quad \sigma^2 = 2, \quad T = 200$ and we look to different simulations for values of ϕ in $[-1, 1]$ where the process is stable. The graph 1 represents the $AR(1)$ process when $\phi = -1$. The second graph, instead, represents the $AR(1)$ process for vaues of ϕ = -0.6, -0.2, 0.2, 0.6, 1. The ligh blue line represent a not stationary process and corresponds to $\phi = -1$. The other processes are similar to the white noise, stationary processes with higher picks when ϕ gets closer to 1.





b) In this step we will fix two different value for ϕ and we will simulate two $AR(1)$ processes (x_1,x_2).

```
x_1=AR1(mu=mu,phi=0.3,T=T,sigma=sigmaS)
x_2=AR1(mu=mu,phi=0.95,T=T,sigma=sigmaS)
```

We now treat the values of μ, ϕ, σ^2 as unknown and we estimate them using MCMC and `RStan`. The following code will sample from the posterior of the three parameters using non-informative priors.

```
#RStan Code:
stanAR1= '
data {
  int<lower=0> N;
  vector[N] x;
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigmaS;
}

model {
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1]-mu), sigmaS);
}'
```

i.) The following output reports info regarding the three inferred parameters when $\phi = 0.3$ (posterior mean=mu and number of effective posterior samples for the parameters).

```
## Inference for Stan model: 894b8b0a524ddf28fede9dc0976eb5ff.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## mu        9.79    0.00 0.14    9.51    9.70    9.78    9.88   10.07  3560
## phi       0.28    0.00 0.07    0.15    0.24    0.28    0.33    0.41  3936
## sigmaS    1.42    0.00 0.07    1.29    1.37    1.42    1.47    1.57  3312
## lp__   -168.66    0.03 1.22 -171.91 -169.21 -168.36 -167.78 -167.29  2115
##        Rhat
## mu        1
## phi       1
## sigmaS    1
## lp__      1
##
## Samples were drawn using NUTS(diag_e) at Mon May 21 12:08:57 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

	CI	mu	sigma	phi
2.5%	L	9.50923	1.293140	1.293140
97.5%	U	10.06814	1.573467	1.573467

Here we have the same kind of output for the three inferred parameters when $\phi = 0.95$

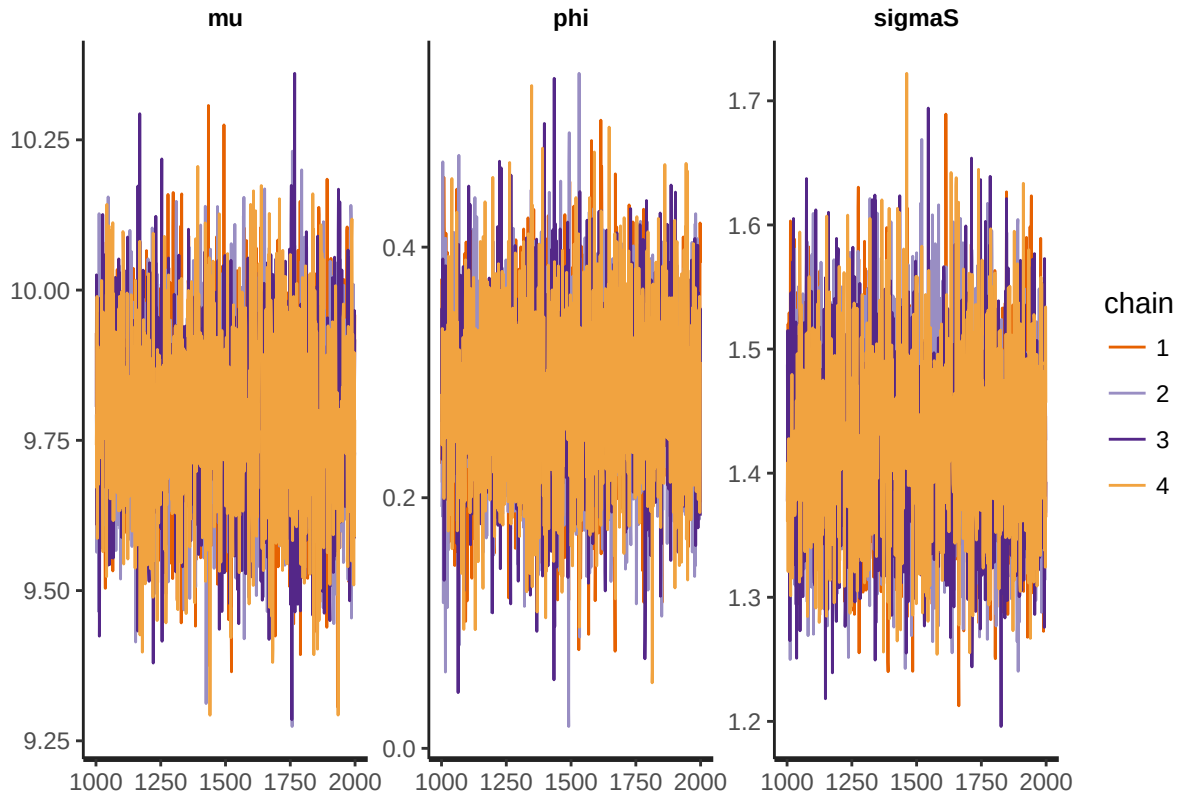
```
## Inference for Stan model: 894b8b0a524ddf28fede9dc0976eb5ff.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## mu       11.54    1.08 5.84    6.24    9.49   10.48   11.52   38.55    29
## phi       0.93    0.00 0.03    0.86    0.90    0.93    0.95    1.00   141
## sigmaS    1.47    0.00 0.08    1.33    1.42    1.47    1.52    1.63   365
## lp__   -175.14    0.16 1.52 -178.91 -175.85 -174.69 -174.01 -173.45    94
##        Rhat
## mu     1.18
## phi    1.04
## sigmaS 1.01
## lp__   1.05
##
## Samples were drawn using NUTS(diag_e) at Mon May 21 12:08:59 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

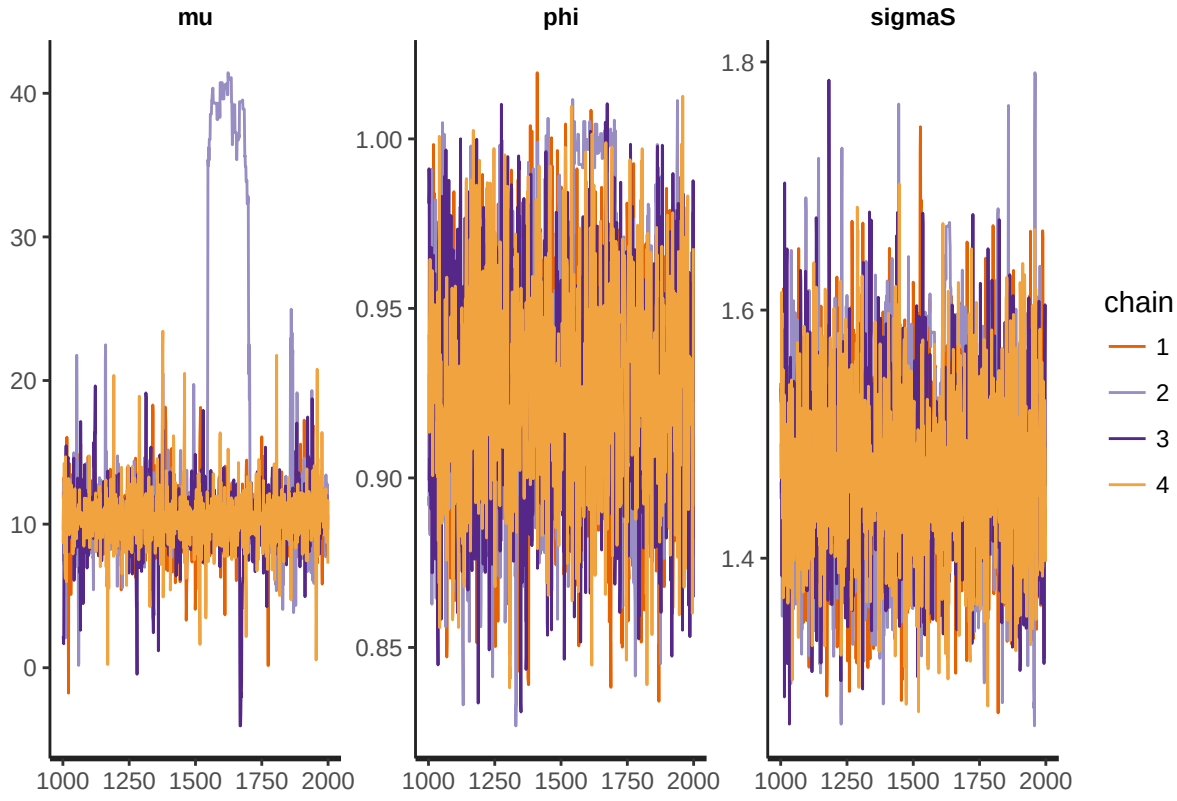
	CI	mu	sigma	phi
2.5%	L	6.23815	1.332971	1.332971
97.5%	U	38.54809	1.626373	1.626373

The posterior mean values are pretty different in the two cases and also the number of effective posterior

samples for the parameters (very high for $\phi = 0.3$ and smaller for $\phi = 0.95$).

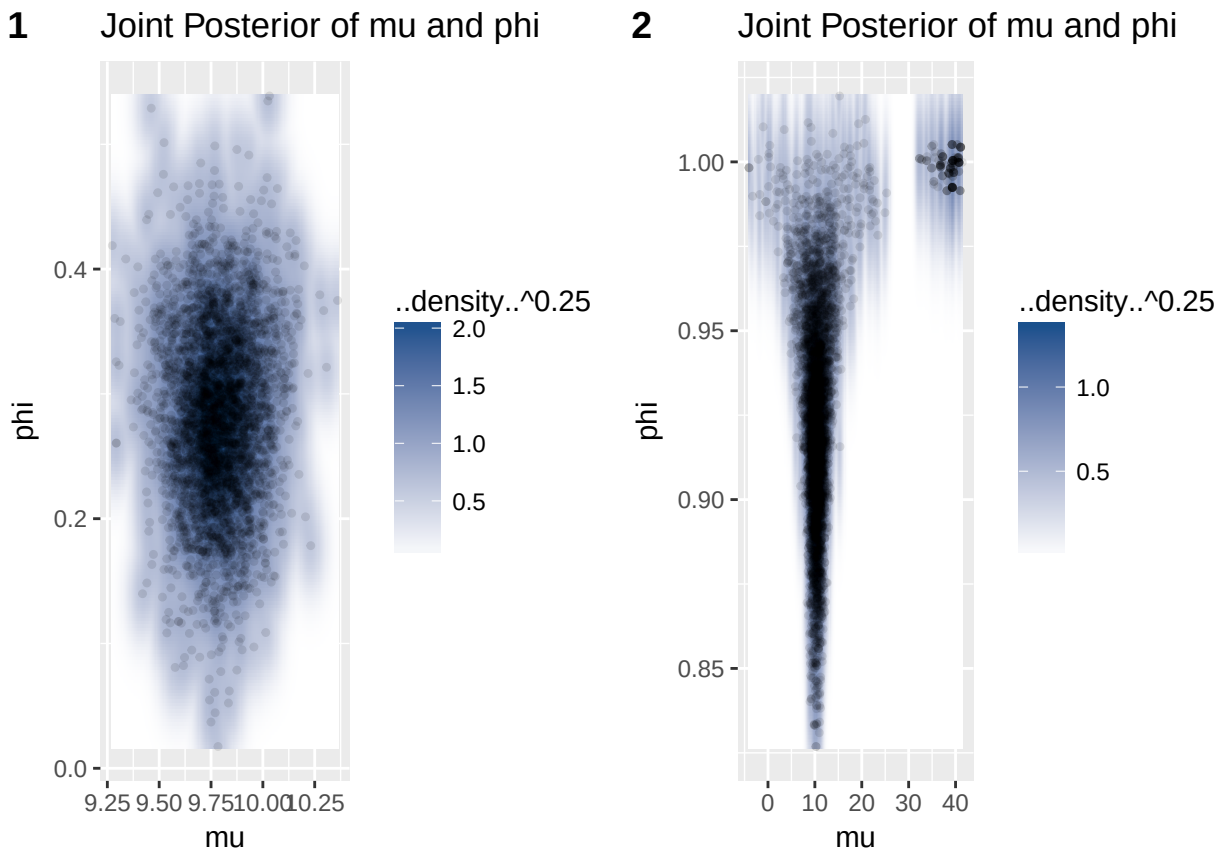
ii.) The following graphs represent the convergence of our parameters.





In both cases we can see that the parameters converge (first plot refers to data where $\phi = .3$, second plot refers to data where $\phi = .95$)

The two following plots show the joint posterior distribution of μ and ϕ.



In case 1. the data were obtained by an AR(1) model with $\phi = 0.3$, in the second case, instead, the model had $\phi = 0.95$. The main difference between the two plots it's the variance,while in the second case the two variables seam to be strongly correlated, in the first case they have a weaker correlation.

c) In this step we will use `campi.dat`, a dataset containing the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. Assuming that the number of infection at each point c_t follows:

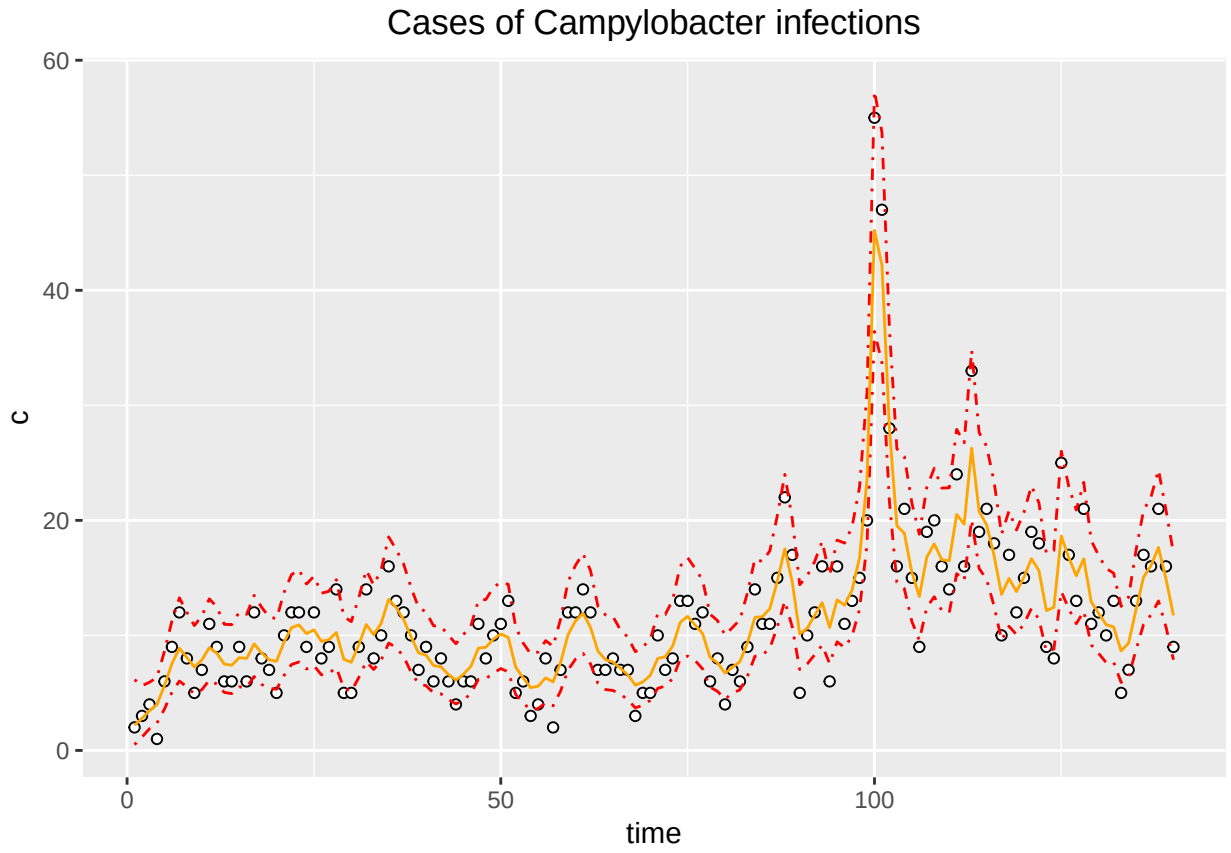
$$c_t|x_t \sim Poisson(\exp(x_t)) \tag{9}$$

where x_t is an AR(1) process as the ones in a), we want to estimate the model in Stan using a non-informative prior.

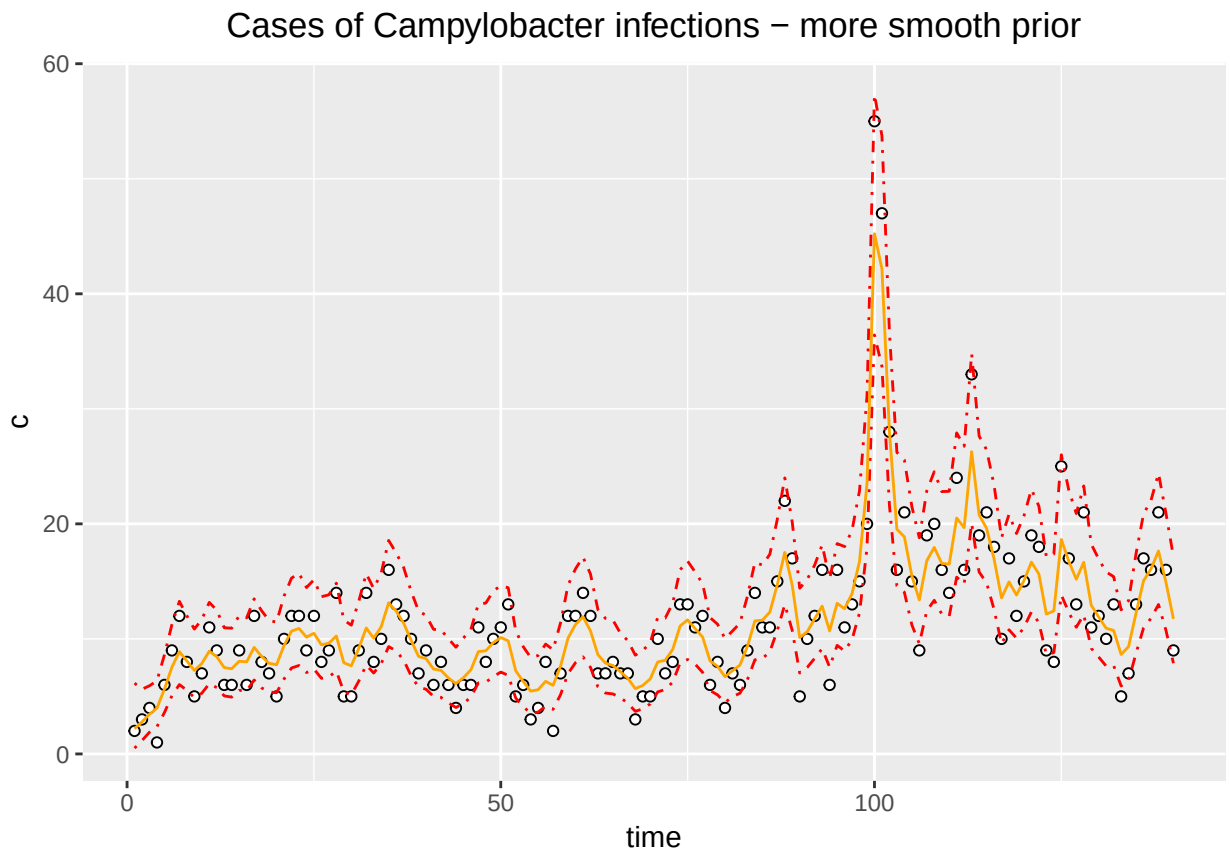
```
c=as.matrix(read.table("campy.dat",header=TRUE))
dimension=dim(c)[1]
campy=list(N=dimension,c=as.vector(c))

#Stan Code
stanPoisson= '
data {
  int<lower=0> N;
  int c[N]; // data model
}
parameters {
  real mu;
  real phi;
  real <lower=0> sigma2;
```

```
  real x[N];
}
transformed parameters {
  real teta[N];
  teta[1]=exp(mu);
  for(n in 2:N)
    teta[n]=exp(x[n]);
}
model {
  for (n in 2:N){
    x[n] ~ normal(mu + phi * (x[n-1]-mu), sqrt(sigma2));
    c[n] ~ poisson(teta[n]); // Poisson
  }

}'
```

The following plot contains the data, the posterior mean (in orange) and the 95% credible intervals for $\theta_t = \exp(x_t)$ over time.



d) In this last step we assume we have a prior belif that the true underlying intensity θ_t varies more smoothly than the data suggests. We chose our prior as $\mu \sim N(0, 10)$ and $\sigma^2 \sim Inv - \chi^2(1, 2)$ and we repeat the same exercise as before.



The graph looks exactly like before, nothing changes adding the prior on σ^2, the prior doesn't affect the posterior in the parameters.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(geoR)
library(ggplot2)
library(ggpubr)
x=as.matrix(read.table("rainfall.dat"))
set.seed(12345)

n=length(x)

#initialization of parameters
mu0=30 #plotting the data the mean value looks closed to around 30
tau0=1
nu0=1
sigma0=1
NIt=10000 #n° iterations

#first value of sigma given
sigma=rinvchisq(1,df=nu0,scale=sigma0)

join=matrix(ncol=2,nrow = NIt)

for(i in 1:NIt){

  w=(n/sigma)/((n/sigma)+(1/tau0))
  mu_n=w*mean(x)+(1-w)*mu0
  tau_n=1/((n/sigma)+(1/tau0)) #it's squared
  nu_n=nu0+n

  mu=rnorm(1,mean=mu_n,sd=tau_n)

  scal=((nu0*sigma0)+sum((x-mu)^2))/(nu_n)
  sigma=rinvchisq(1,df=nu_n,scale=sqrt(scal))

  join[i,1]=mu
  join[i,2]=sigma

}

mean_posterior=data.frame(mu=join[,1],time=seq(1,length(join[,1])))
sigma_posterior=data.frame(sigma=join[,2],time=seq(1,length(join[,2])))

a=ggplot(data=mean_posterior,aes(x=mu))+geom_histogram(aes(y=..density..),colour = "black", fill = "whi

b=ggplot(data=sigma_posterior,aes(x=sigma))+geom_histogram(aes(y=..density..),colour = "black", fill =

ggarrange(a,b,
          labels = c("1", "2"),
          ncol = 2, nrow = 1)


joint_posterior=data.frame(mu=join[,1],sigma=join[,2])
```

```r
ggplot(data=joint_posterior,aes(x=mu,y=sigma))+geom_point(data=joint_posterior,aes(x=mu,y=sigma),colour


c=ggplot()+geom_path(data=mean_posterior,aes(x=time,y=mu))+labs(title="Convergence of mu")+theme(plot.ti

d=ggplot()+geom_path(data=sigma_posterior,aes(x=time,y=sigma))+labs(title="Convergence of sigma")+theme

ggarrange(c,d,
          labels = c("1", "2"),
          ncol = 1, nrow = 2)

nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2

nIter <- 1000

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.1 # Adding sleep time between iterations for plotting

#Function that simulates from the
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

#Function that simulates from a Dirichlet distribution
rDirichlet <- function(param){
  nCat <- length(param)
  piDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    piDraws[j] <- rgamma(1,param[j],1)
  }
  piDraws = piDraws/sum(piDraws)
  return(piDraws)
}

# Simple function that converts between two different representations of the mixture allocation
S2alloc <- function(S){
  n <- dim(S)[1]
  alloc <- rep(0,n)
  for (i in 1:n){
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}
```

```r
# Initial value
nObs <- length(x)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp)))
mu <- quantile(x, probs = seq(0,1,length = nComp))
sigma2 <- rep(var(x),nComp)
probObsInComp <- rep(NA, nComp)



# Setting up the plot
xGrid <- seq(min(x)-1*apply(x,2,sd),max(x)+1*apply(x,2,sd),length = 100)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean <- rep(0,length(xGrid))
effIterCount <- 0
ylim <- c(0,2*max(hist(x)$density))
rainfall=data.frame(x=x)
hist=ggplot(data=rainfall,aes(x=x))+geom_histogram(aes(y=..density..),colour = "black", fill = "white",

hi=hist


for (k in 1:nIter){
  #message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the group al
  nAlloc <- colSums(S)
  #print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j], scale = (nu0[j]*sigma2_0[j] + sum((x[alloc =
  }

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }
```

```
  # Printing the fitted density against data histogram
  if (plotFit && (k%%100 ==0)){
    effIterCount <- effIterCount + 1
    hi=hist
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens

      da=data.frame(xGrid=xGrid,compDens=compDens)

      hi=hi+geom_path(data=da,aes(x=xGrid,y=compDens),col = lineColors[j])

      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    da2=data.frame(xGrid=xGrid,mixDens=mixDens)

    hi= hi+geom_path(data=da2,aes(x=xGrid,y=mixDens),col = "red")

  }

}


hi+labs(title="Mixture Normal model")+theme(plot.title = element_text(hjust = 0.5))



a_point=data.frame(x=xGrid,y=dnorm(xGrid,mean=mean(join[,1]),sd=mean(join[,2])))
b_point=data.frame(x=xGrid,y=mixDensMean,col="blue")

hist+geom_density(colour = "red",adjust=3)+geom_path(data=b_point,aes(x=x,y=y),col="blue")+geom_path(da


library(ggplot2)
library(ggpubr)
library(knitr)
library(kableExtra)
library(rstan)

mu=10
sigmaS=2
T=200
phi=seq(-1,1,.4) #values of phi
set.seed(12345)

AR1=function(mu,phi,sigma,T){
  x<-vector()
  x[1]=mu
  for(i in 2:T){
```

```
    eps=rnorm(1,mean=0,sd=sqrt(sigma))
    x[i]=mu+phi*(x[i-1]-mu)+eps
  }
  return(data.frame(x=seq(1:length(x)),series=as.vector(x)))
}
set.seed(12345)

x=AR1(mu=mu,phi=phi[1],T=T,sigma=sigmaS)
p=ggplot()+geom_path(data=x,aes(y=series,x=x))+ggtitle(paste("AR1 process with phi=-1"))

q=ggplot()

for(i in 2:length(phi)){
  x=AR1(mu=mu,phi=phi[i],T=T,sigma=sigmaS)
  q=q+geom_path(data=x,aes(y=series,x=x),col=i+7)
}

q=q+ggtitle("AR1 process with different values of phi")

ggarrange(p, q,
          labels = c("1", "2"),
          ncol = 1, nrow = 2)

x_1=AR1(mu=mu,phi=0.3,T=T,sigma=sigmaS)
x_2=AR1(mu=mu,phi=0.95,T=T,sigma=sigmaS)
#RStan Code:
stanAR1= '
data {
  int<lower=0> N;
  vector[N] x;
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigmaS;
}

model {
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1]-mu), sigmaS);
}'
#Our data: phi=.3
fitModel_3<-stan(model_code=stanAR1,
           data=list(N=200,x=x_1$series),
           par=c("mu","phi","sigmaS"),
           warmup=1000,
           iter=2000,
           chains=4)
#Our data: phi=.95
fitModel_95<-stan(model_code=stanAR1,
           data=list(N=200,x=x_2$series),
           par=c("mu","phi","sigmaS"),
           warmup=1000,
```

```
            iter=2000,
            chains=4)
print(fitModel_3,digits=2)

res_3<-rstan::extract(fitModel_3)

CI_3_mu=quantile(res_3$mu, c(0.025, 0.975))
CI_3_sigma=quantile(res_3$sigmaS, c(0.025, 0.975))
CI_3_phi=quantile(res_3$phi, c(0.025, 0.975))

tab=data.frame(CI=c("L","U"),mu=CI_3_mu,sigma=CI_3_sigma,phi=CI_3_sigma)

kable(tab)

#traceplot(fitModel3)

#pairs(fitModel)

print(fitModel_95,digits=2)

res_95<-rstan::extract(fitModel_95)

CI_95_mu=quantile(res_95$mu, c(0.025, 0.975))
CI_95_sigma=quantile(res_95$sigmaS, c(0.025, 0.975))
CI_95_phi=quantile(res_95$phi, c(0.025, 0.975))

tab2=data.frame(CI=c("L","U"),mu=CI_95_mu,sigma=CI_95_sigma,phi=CI_95_sigma)

kable(tab2)

traceplot(fitModel_3)
traceplot(fitModel_95)
scatt_3=data.frame(mu=res_3$mu,phi=res_3$phi)
scatt_95=data.frame(mu=res_95$mu,phi=res_95$phi)

scat_3=ggplot(data = scatt_3, aes(x=mu, y=phi)) +   stat_density2d(aes(fill = ..density..^0.25), geom =

scat_95=ggplot(data = scatt_95, aes(x=mu, y=phi)) +   stat_density2d(aes(fill = ..density..^0.25), geom

ggarrange(scat_3, scat_95,
          labels = c("1", "2"),
          ncol = 2, nrow = 1)


c=as.matrix(read.table("campy.dat",header=TRUE))
dimension=dim(c)[1]
campy=list(N=dimension,c=as.vector(c))

#Stan Code
stanPoisson= '
data {
  int<lower=0> N;
  int c[N]; // data model
```

```
}
parameters {
  real mu;
  real phi;
  real <lower=0> sigma2;
  real x[N];
}
transformed parameters {
  real teta[N];
  teta[1]=exp(mu);
  for(n in 2:N)
    teta[n]=exp(x[n]);
}
model {
  for (n in 2:N){
    x[n] ~ normal(mu + phi * (x[n-1]-mu), sqrt(sigma2));
    c[n] ~ poisson(teta[n]); // Poisson
  }

}'
fitData=stan(model_code = stanPoisson,
             data=campy,
             warmup = 1000,
             iter=2000,
             chains=4)

res<-extract(fitData)

lowCI<-vector()
highCI<-vector()
MEAN<-vector()

#95% credible interval
for(i in 1:dimension){

  data=exp(res$x[,i])
  CI=quantile(data, c(0.05, 0.975))
  lowCI[i]=CI[1]
  highCI[i]=CI[2]
  MEAN[i]=mean(data)

}

camp=data.frame(c=as.vector(c),time=seq(1,length(c)),LCI=lowCI,UCI=highCI,Mean=MEAN)

ggplot()+geom_point(data=camp,aes(x=time,y=c),colour = "black", fill = "white",shape=21)+labs(title="Cas


c=as.matrix(read.table("campy.dat",header=TRUE))
dimension=dim(c)[1]


campy=list(N=dimension,c=as.vector(c))
```

```
stanPoisson2= '
data {
  int<lower=0> N;
  int c[N]; // data model
}
parameters {
  real mu;
  real phi;
  real <lower=0> sigma2;
  real x[N];
}
transformed parameters {
  real teta[N];
  teta[1]=exp(mu);
  for(n in 2:N)
    teta[n]=exp(x[n]);
}
model {
  mu ~ normal(0,10); // Normal with mean 0, st.dev. 100
  sigma2 ~ scaled_inv_chi_square(100,.1); // strong prior with sigma2 very small
  for (n in 2:N){
    x[n] ~ normal(mu + phi * (x[n-1]-mu), sqrt(sigma2));
    c[n] ~ poisson(teta[n]); // Poisson
  }

}'

fitData2=stan(model_code = stanPoisson2,
              data=campy,
              warmup = 1000,
              iter=2000,
              chains=4)

print(fitData2,digits_summary =3)

res2<-extract(fitData2)
lowCI2<-vector()
highCI2<-vector()
MEAN2<-vector()

#95% credible interval
for(i in 1:dimension){

  data=exp(res2$x[,i])
  CI=quantile(data, c(0.05, 0.975))
  lowCI2[i]=CI[1]
  highCI2[i]=CI[2]
  MEAN2[i]=mean(data)

}

camp2=data.frame(c=as.vector(c),time=seq(1,length(c)),LCI=lowCI2,UCI=highCI2,Mean=MEAN2)
```

```
ggplot()+geom_point(data=camp2,aes(x=time,y=c),colour = "black", fill = "white",shape=21)+labs(title="Ca
```

Lab 4 - Bayesian Learning

Alessia De Biase and Alejandro Garcia

15 maggio 2018

Assignment 1: Poisson regression - the MCMC way

In this assignment we will use the dataset `eBayNumberOfBidderData.dat` which contains observations from 1000 eBay auctions of coins. The response variable in the model is `nBids` which records the number of bids in each auction. The model we consider is the following:

$$y_i|\beta \sim Poisson[\exp(\mathbf{x}_i^T\beta)], \quad i = 1, \ldots, n \tag{1}$$

a) We first use the `glm()` function to obtain the maximum likelihood estimator of β in the Poisson regression model (we specify `family=poisson()`) and we check which of the covariates are significant.

```
model=glm(nBids~PowerSeller+VerifyID+Sealed+Minblem+MajBlem+LargNeg+LogBook+MinBidShare,
          data=ebay_data,family=poisson())
summary(model)
```

```
##
## Call:
## glm(formula = nBids ~ PowerSeller + VerifyID + Sealed + Minblem +
##     MajBlem + LargNeg + LogBook + MinBidShare, family = poisson(),
##     data = ebay_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5800  -0.7222  -0.0441   0.5269   2.4605
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.07244    0.03077  34.848  < 2e-16 ***
## PowerSeller -0.02054    0.03678  -0.558   0.5765
## VerifyID    -0.39452    0.09243  -4.268 1.97e-05 ***
## Sealed       0.44384    0.05056   8.778  < 2e-16 ***
## Minblem     -0.05220    0.06020  -0.867   0.3859
## MajBlem     -0.22087    0.09144  -2.416   0.0157 *
## LargNeg      0.07067    0.05633   1.255   0.2096
## LogBook     -0.12068    0.02896  -4.166 3.09e-05 ***
## MinBidShare -1.89410    0.07124 -26.588  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2151.28  on 999  degrees of freedom
## Residual deviance:  867.47  on 991  degrees of freedom
## AIC: 3610.3
##
## Number of Fisher Scoring iterations: 5
```

The most significant covariates according to our model are: `VerifyID`, `Sealed`, `LogBook` and `MinBidShare`.

b) In this step we do a Bayesian analysis of the Poisson regression. Let's assume first that the posterior density is approximately multivariate normal:

$$\beta|y \sim N(\tilde{\beta}, J_{\mathbf{y}}^{-1}(\tilde{\beta})) \tag{2}$$

where $\tilde{\beta}$ is the posterior mode and $J(\tilde{\beta})$ is the observed Hessian evaluated at the posterior mode.

The prior we will use is called Zellner's g-prior and is:

$$\beta \sim N[\mathbf{0}, 100(\mathbf{X}^T\mathbf{X})^{-1}] \tag{3}$$

where $\mathbf{X}$ is the n x p covariate matrix.

Let's call $\lambda_i = \exp(\mathbf{x}_i^T\beta)$ so the likelihood function in this case is:

$$p(Y_i = y|X_i, \beta) = \frac{e^{-\lambda_i}\lambda_i^y}{y!} \tag{4}$$

$$L(\beta|x) = p(\mathbf{Y}|\mathbf{X};\beta) = \prod_i p(y_i|x_i,\beta) = \prod_i \frac{\exp(-\exp(\mathbf{x}_i^T\beta))(\exp(\mathbf{x}_i^T\beta))^y}{y!} = \prod_i \frac{\exp(-\exp(\mathbf{x}_i^T\beta) + \mathbf{x}_i^T\beta y}{y!} \tag{5}$$

The logLikelihood function is:

$$logL(\beta|x) = \log(\prod_i \frac{\exp(-\exp(\mathbf{x}_i^T\beta) + \mathbf{x}_i^T\beta y}{y!}) = \sum_i (\mathbf{x}_i^T\beta y - \exp(\mathbf{x}_i^T\beta) - \log(y!)) \tag{6}$$

The posterior we want to maximize is:

$$p(\beta|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X};\beta)p(\beta) \tag{7}$$

The logarithm of the posterior will be then proportional to the sum of the logarithms of the prior and the likelihood so in our code we will use the function `LogPostPoisson` which will return the sum of them.

```
# Prior:
nPara=dim(x)[2]


#Post LogLikelihood
LogPostPoisson <- function(betaVect,y,X){
  dim <- length(betaVect);
  sigma=100*(t(X)%*%X);
  linPred <- X%*%betaVect;
  logLik <- sum( linPred*y - exp(linPred)-log(factorial(y)));
  if (abs(logLik) == Inf) logLik = -20000;
  logPrior <- dmvnorm(betaVect, matrix(0,dim,1), sigma, log=TRUE);
  return(logLik + logPrior)
}

start <- as.vector(rep(0,nPara));

Optim<-optim(start,LogPostPoisson,gr=NULL,y,x,method=c("BFGS"),control=list(fnscale=-1),
```

```
              hessian=TRUE)

approxPostStd <- sqrt(diag(-solve(Optim$hessian)))
approxPostMean <- round(Optim$par,2)
```

Using the function `optim` we obtain that the posterior mode is 1.07, -0.02, -0.39, 0.44, -0.05, -0.22, 0.07, -0.12, -1.89 and the approximate posterior standard deviation is 0.03, 0.04, 0.09, 0.05, 0.06, 0.09, 0.06, 0.03, 0.07 .

c) In this step we will simulate from the actual posterior of β using the Metropolis algorithm. We want to program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density.

The *proposal density* is the multivariate normal density:

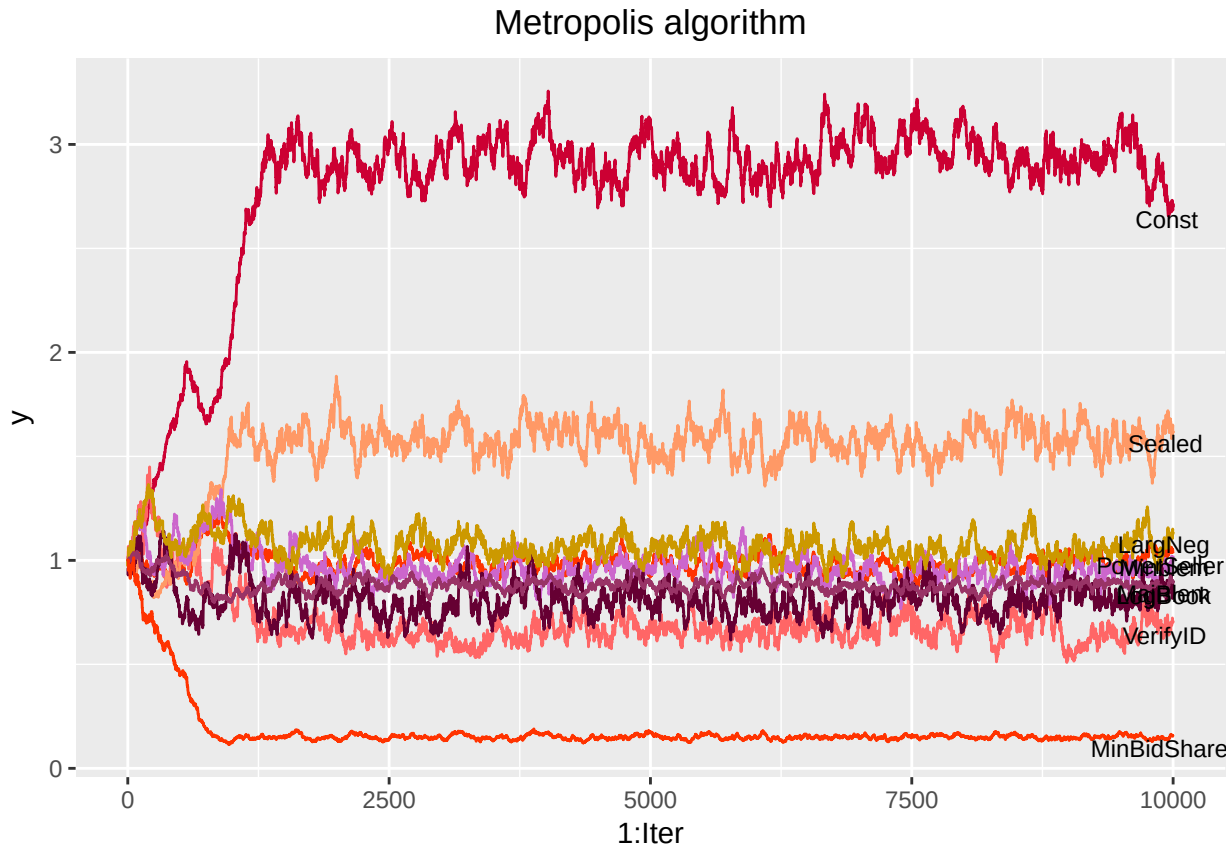
$$\theta_p|\theta_c \sim N(\theta_c, \tilde{c} \cdot \Sigma) \tag{8}$$

where Σ is the inverse of the observed Hessian evaluated at the posterior mode obtained in b) ($\Sigma = J_y^{-1}(\tilde{\beta})$) and θ_c is the current draw.

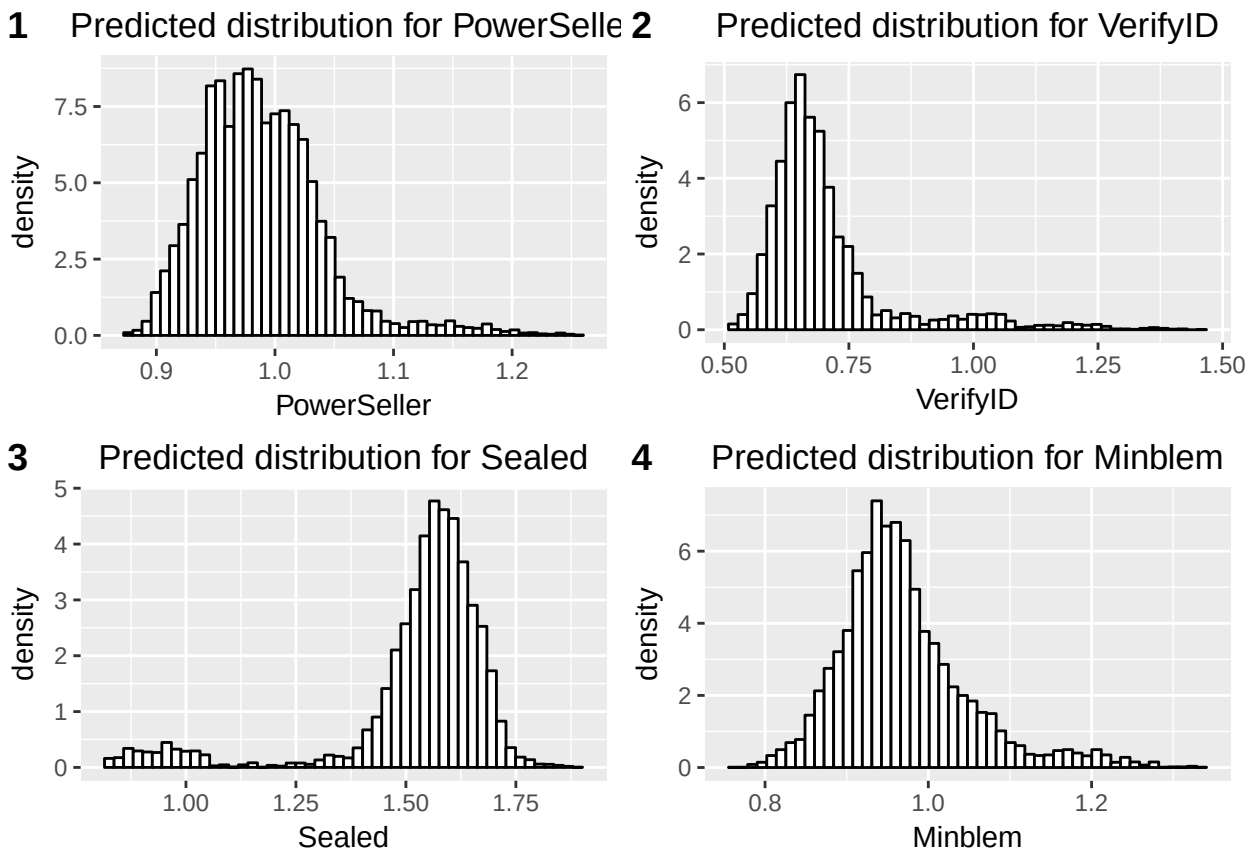
The following is the implementation of the algorithm:

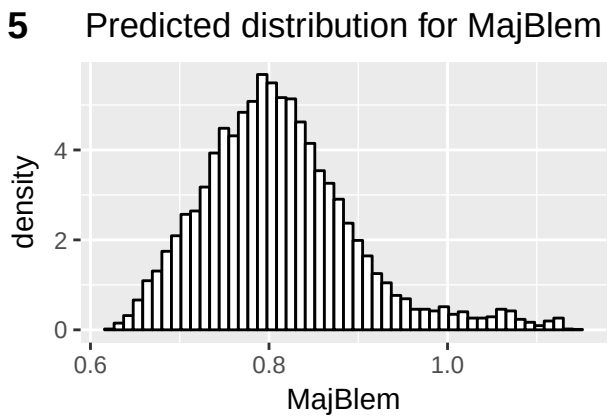
```
# logPostFunc(theta): function object that computes the
#                     log posterior density at any value
#                     of the parameter vector theta

metropolis=function(logPostFunc,theta,c,Sigma,nIter=1000,X,y...){

  set.seed(12345)
  out=matrix(ncol=length(theta),nrow = nIter)
  count=0
  for(i in 1:nIter){

    #new vector of parameters from the proposal:
    new=as.vector(rmvnorm(1,mean=theta,sigma=c*Sigma))

    #acceptance probability:
    ratio=exp(logPostFunc(new,y=y,X=X)-logPostFunc(theta,y=y,X=X))
    if(runif(1)<min(1,ratio))  {theta=new; count=count+1;}

    out[i,]<-theta
  }
  rej=count/nIter

  return(list(parameters=out,avg_rej=rej))

}

Iter=10000

output=metropolis(logPostFunc=LogPostPoisson,theta=start,c=0.05,Sigma=-solve(Optim$hessian),
                  y=y,X=x,nIter=Iter)

result=output$parameters

rej=100-output$avg_rej*100
```

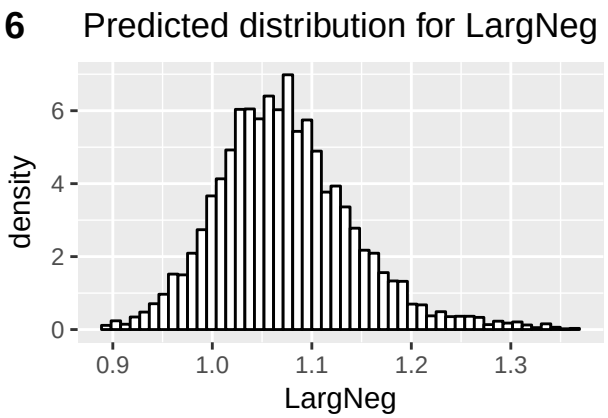
The following plot shows the convergence of our variables. We consider $\phi_j = \exp(\beta_j)$ instead of simply β_j because it's easier to interpret. The graph is very sensible to the change of the parameter $\tilde{c}$. We choose $\tilde{c} = 0.05$ because with this result we obtain an average rejecting probability of 27.89 % which is in the range 25-30%.

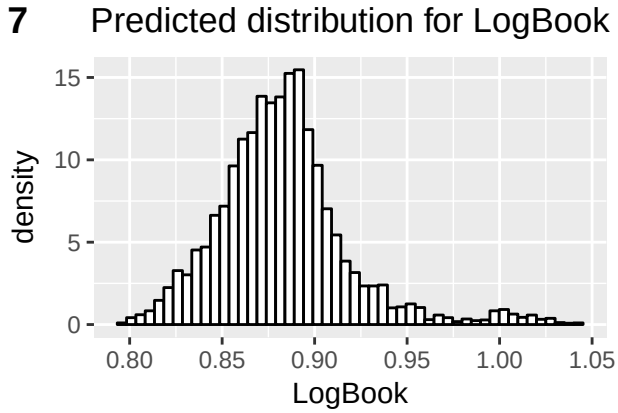


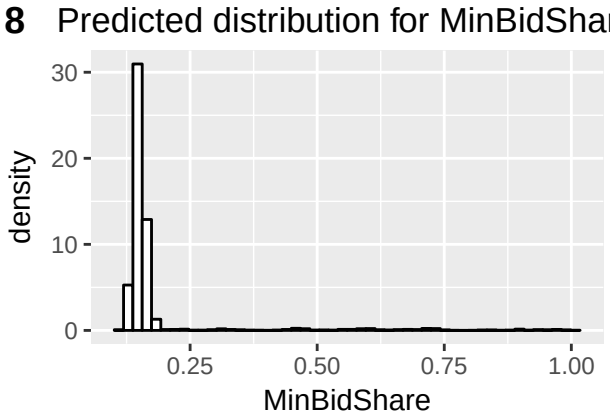
The following plots represent the posterior distribution of ϕ_j for all the variables.


1
Predicted distribution for PowerSelle
density
PowerSeller
2
Predicted distribution for VerifyID
density
VerifyID
3
Predicted distribution for Sealed
density
Sealed
4
Predicted distribution for Minblem
density
Minblem





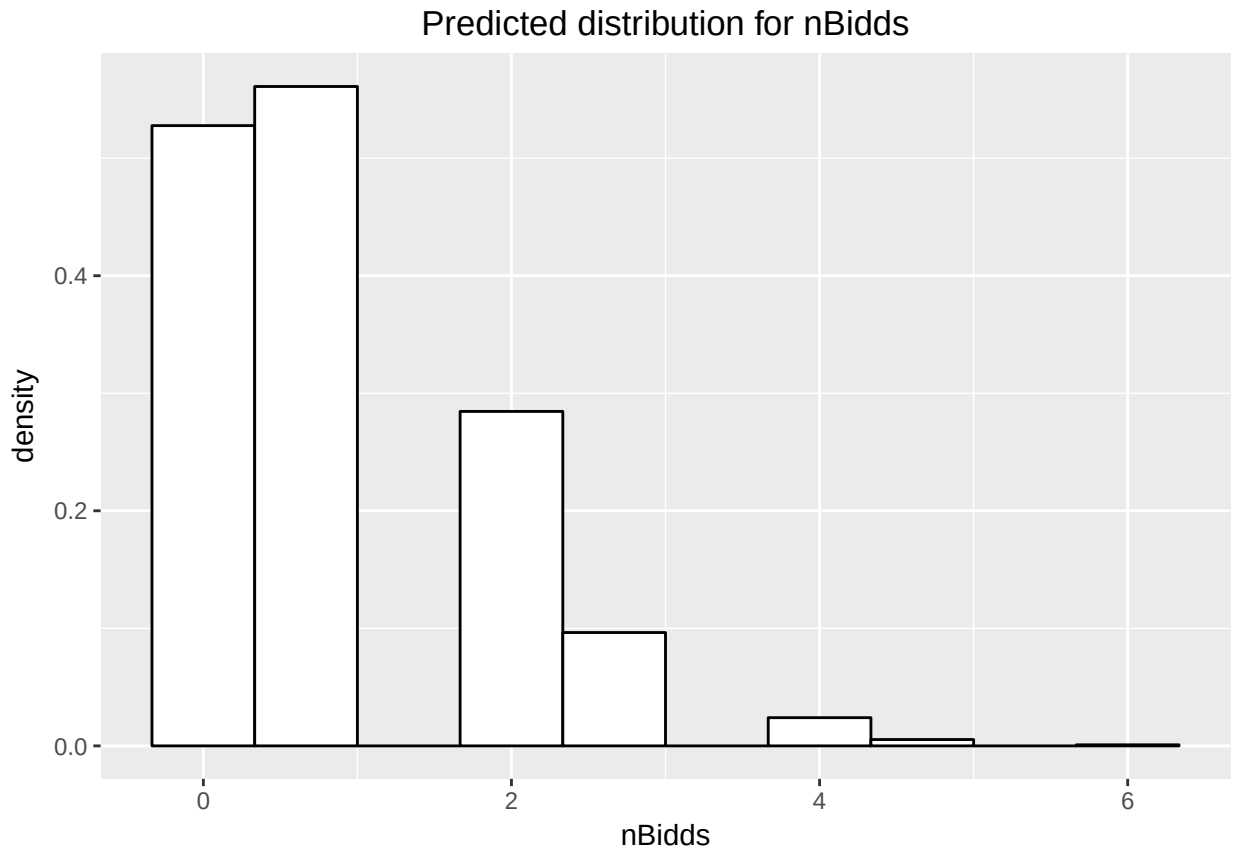




d) In this last step we use the draws from c) to simulate from the predictive distribution of the number of bidders in a new auction such that:

- **PowerSeller**=1
- **VerifyID**=1
- **Sealed**=1
- **MinBlem**=0
- **MajBlem**=0
- **LargNeg**=0
- **LogBook**=1
- **MinBidShare**=0.5

The following plot represents the predictive distribution of the number of bidders in the new auction.



The probability of no bidders in this new auction is: 35.18 %.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(glmnet)
library(mvtnorm)
library(ggpubr)

ebay_data=as.data.frame(read.table("eBayNumberOfBidderData.dat",header=TRUE))


y=ebay_data$nBids
x=as.matrix(ebay_data[,2:dim(ebay_data)[2]])
variables=colnames(x)

model=glm(nBids~PowerSeller+VerifyID+Sealed+Minblem+MajBlem+LargNeg+LogBook+MinBidShare,
          data=ebay_data,family=poisson())
summary(model)


# Prior:
nPara=dim(x)[2]


#Post LogLikelihood
LogPostPoisson <- function(betaVect,y,X){
  dim <- length(betaVect);
  sigma=100*(t(X)%*%X);
  linPred <- X%*%betaVect;
  logLik <- sum( linPred*y - exp(linPred)-log(factorial(y)));
  if (abs(logLik) == Inf) logLik = -20000;
  logPrior <- dmvnorm(betaVect, matrix(0,dim,1), sigma, log=TRUE);
  return(logLik + logPrior)
}

start <- as.vector(rep(0,nPara));

Optim<-optim(start,LogPostPoisson,gr=NULL,y,x,method=c("BFGS"),control=list(fnscale=-1),
             hessian=TRUE)

approxPostStd <- sqrt(diag(-solve(Optim$hessian)))
approxPostMean <- round(Optim$par,2)


# logPostFunc(theta): function object that computes the
#                     log posterior density at any value
#                     of the parameter vector theta

metropolis=function(logPostFunc,theta,c,Sigma,nIter=1000,X,y...){

  set.seed(12345)
  out=matrix(ncol=length(theta),nrow = nIter)
  count=0
```

```
  for(i in 1:nIter){

    #new vector of parameters from the proposal:
    new=as.vector(rmvnorm(1,mean=theta,sigma=c*Sigma))

    #acceptance probability:
    ratio=exp(logPostFunc(new,y=y,X=X)-logPostFunc(theta,y=y,X=X))
    if(runif(1)<min(1,ratio))  {theta=new; count=count+1;}

    out[i,]<-theta
  }
  rej=count/nIter

  return(list(parameters=out,avg_rej=rej))

}

Iter=10000

output=metropolis(logPostFunc=LogPostPoisson,theta=start,c=0.05,Sigma=-solve(Optim$hessian),
                  y=y,X=x,nIter=Iter)

result=output$parameters

rej=100-output$avg_rej*100

result_exp=as.data.frame(exp(result))
colnames(result_exp)<-variables

ggplot(data=result_exp,aes(x=1:Iter))+geom_line(aes(y=Const),col="#CC0033")+geom_line(aes(y=PowerSeller)

a=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=PowerSeller),colour = "black", fill = "whi

b=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=VerifyID),colour = "black", fill = "white"

c=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=Sealed),colour = "black", fill = "white",bi

d=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=Minblem),colour = "black", fill = "white",b

e=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=MajBlem),colour = "black", fill = "white",b

f=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=LargNeg),colour = "black", fill = "white",b

g=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=LogBook),colour = "black", fill = "white",b

h=ggplot(data=result_exp)+geom_histogram(aes(y=..density..,x=MinBidShare),colour = "black", fill = "whi

ggarrange(a,b,c,d,
          labels = c("1", "2","3","4"),
          ncol = 2, nrow = 2)

ggarrange(e,f,g,h,
          labels = c("5","6","7","8"),
```

```
          ncol = 2, nrow = 2)


new_obs<-matrix(c(1,1,1,1,0,0,0,1,0.5),nrow = 9,ncol=1) #new auction plus constant

lambda=exp(as.matrix(result)%*%new_obs)
y_new<-vector()

for(i in 1:length(lambda)){
  y_new[i]=rpois(1,lambda[i])
}

prob=sum(y_new==0)/Iter*100

y_new=as.data.frame(y_new)

colnames(y_new)="nBidds"

ggplot(data=y_new)+geom_histogram(aes(y=..density..,x=nBidds),colour = "black", fill = "white",bins=10)+
```