# Lab1

*Naveen Gabriel ,Sridhar Adhikarla*

*2019-04-14*

## Contents

**1 1 Bernoulli ... again.** **2**
1.1 a.) Draw random numbers from the Beta posterior . . . 2
1.2 b.) Compute the posterior probability $Pr(\theta < 0.4|y)$ . . . 3
1.3 c.)Posterior distribution of the log-odds . . . 3

**2 2. Log-normal distribution and the Gini coefficient.** **4**
2.1 a.) Simulate 10000 draws from the posterior of $\sigma^2$ . . . 4
2.2 b.) Gini coeffecient . . . 5
2.3 c.)Equal tailed interval vs HPD of posterior. . . . 6

**3 3. Von Mises distribution.** **7**

**4 Appendix** **10**



# 1 1 Bernoulli ... again.

We have Beta posterior distribution as :

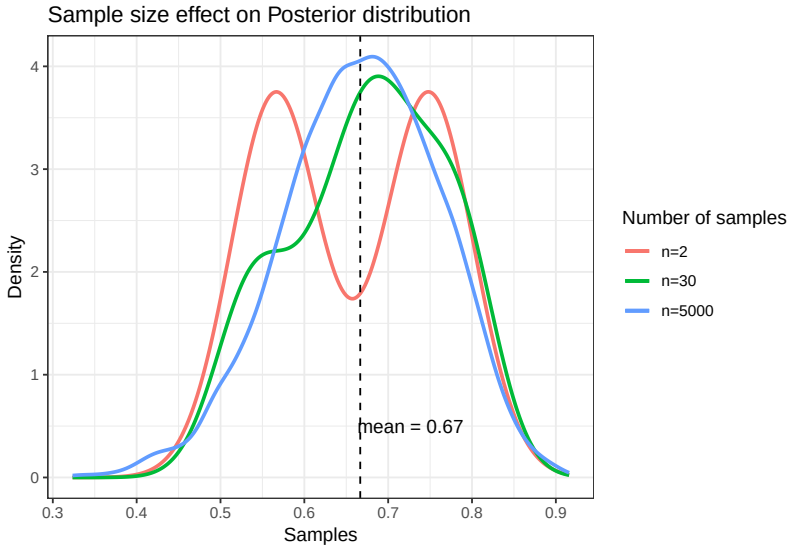
$$\theta|y \sim Beta(\alpha_0 + s, \beta_0 + f)$$
$$\theta|y \sim Beta(16, 8)$$

## 1.1 a.) Draw random numbers from the Beta posterior

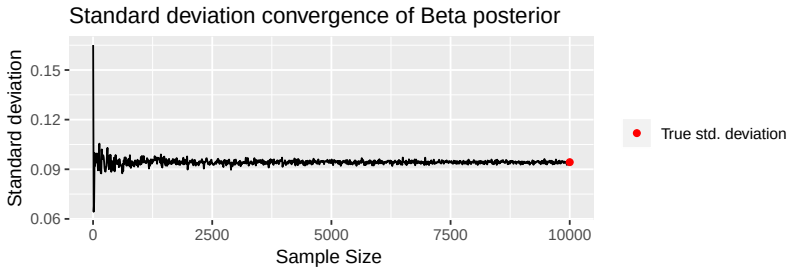
Knowing the parameters of posterior distribution, we computed the true mean and variance distribution using the below formula:

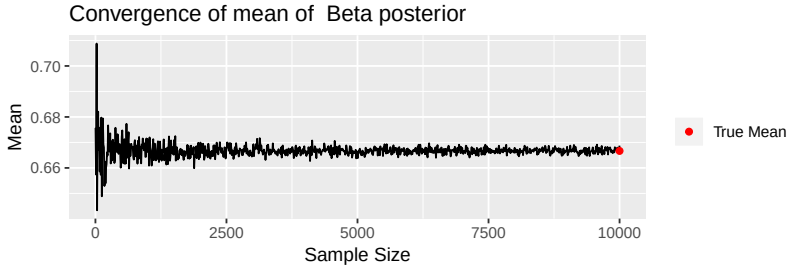
$$\mu = \frac{\alpha}{\alpha + \beta}$$
$$\sigma^2 = \frac{\alpha * \beta}{(\alpha + \beta)^2.(\alpha + \beta + 1)}$$

Plotting the density graph shows how most of the numbers in sample centered aroudn true mean as the number of samples is increased.



Below graphs shows the convergence of mean and standard deviation to actual mean and standard deviation as the number of samples increases.




### 1.2 b.) Compute the posterior probability $Pr(\theta < 0.4|y)$

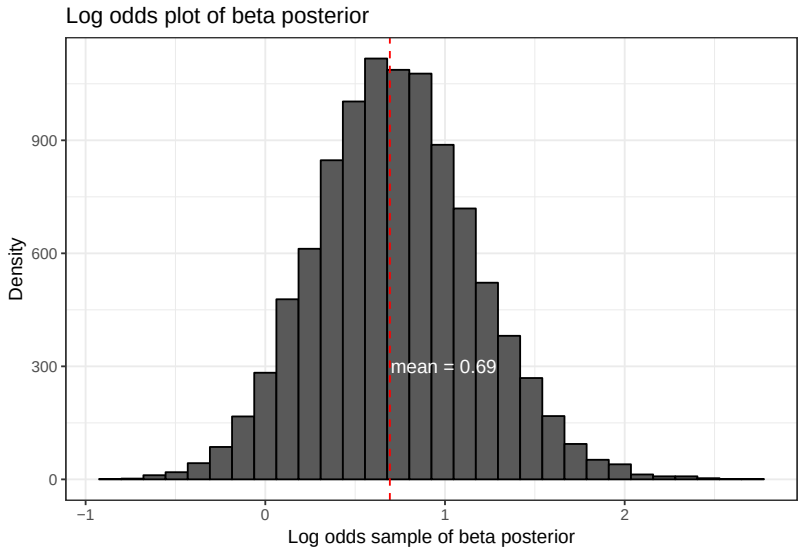
We sampled 10000 points from Beta posterior and checked how many of those points has value lesser than 0.4. Below are computed and true results of $Pr(\theta < 0.4|y)$. Both of these values are nearly same.

```
Computed value of CDF : 0.004


True value of CDF     : 0.00397
```

### 1.3 c.)Posterior distribution of the log-odds

Log odds are an alternate way of expressing probabilities, which simplifies the process of updating them with new evidence. Since our posterior samples are in probabilities, we convert them into log odds and check if the new mean of samples is nearly same as the transformed mean of posterior which is shown as red line. We conclude that, even after tranforming posterior parameters to log odds, it mean of the sample agrees with the sam-






ples generated.

## 2 2. Log-normal distribution and the Gini coefficient.

### 2.1 a.) Simulate 10000 draws from the posterior of $\sigma^2$

The task was to simulate 10,000 draws of $\sigma^2$ from posterior distribution and compare it with the theoretical chi square distribution.

Model:

$$logx_1, logx_2.....logx_n \sim \mathcal{N}(\mu, \sigma^2)$$

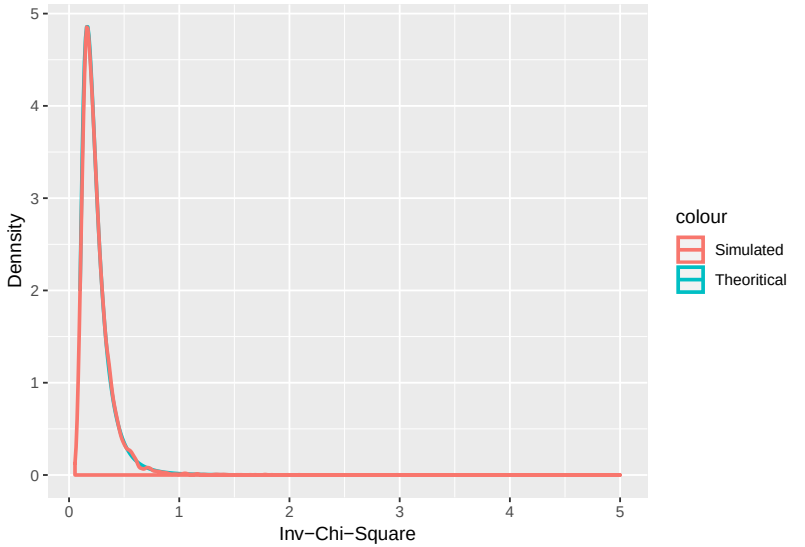
Non informative Prior:

$$p(\sigma^2) \propto (\sigma^2)^{-1}$$

Posterior :

$$\sigma^2|x \sim Inv - \chi(n-1, s^2)$$

We used the pdf of scaled inverse chi square to draw samples for our theoretical value. After drawing samples from both we plotted them as below.




The red curve shows our 10,000 draw from posterior distribution. It gives an idea about probable values of chi square which we can accept. From our posterior it seems most of the chi sqaure value is around 0.1 which we may accept for further computation.

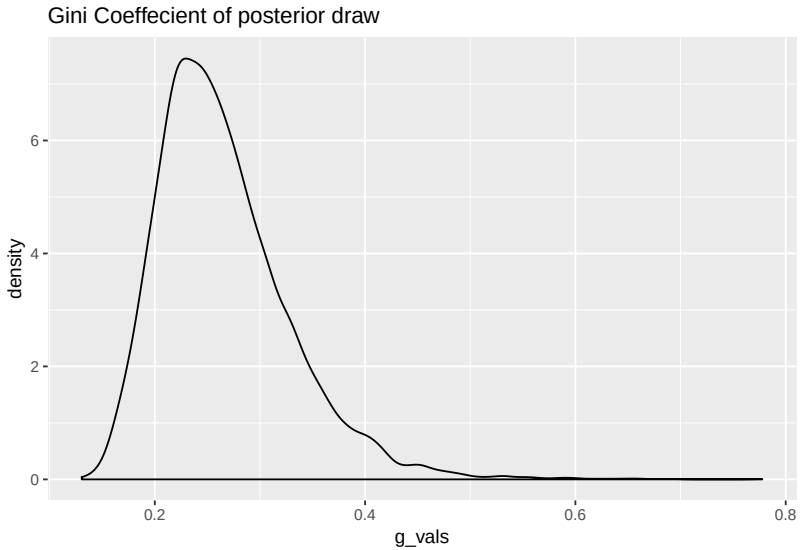
Blue colour shows the theorectical chi square and it seems to be similar to the distribution of posterior. Though both is not accurate to dot but we can say our posterior has done good job in estimating parameter which resembles nearly to the theoretical inverse chi sqaure distribution.

### 2.2 b.) Gini coeffecient

We have salary (in thousand SEK ) from the following ten observations: 14, 25, 45, 25, 30, 33, 19, 50, 34 and 67. Gini coefficient of 0 expresses perfect equality, where all values are the same (where everyone has the same income). A Gini coefficient of 1 (or 100%) expresses maximal inequality among value .

Based on our previous draw of $\sigma^2$ from posterior we computed the gini coeffiecient. Below graph shows the probable value of gini value. From the graph we can immediately say that most of the value is nealry 0.3 hence there is high chances that the mentioned salary is nearly same among the people.





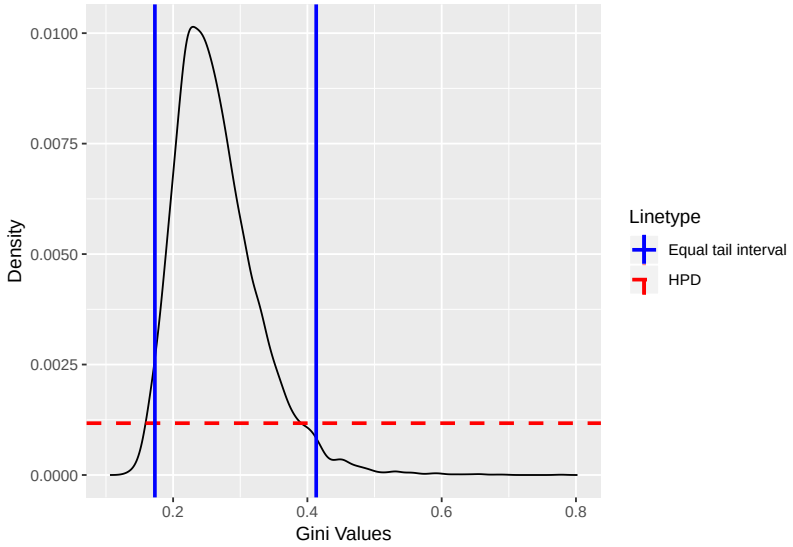

### 2.3 c.)Equal tailed interval vs HPD of posterior.

Using the posterior draws from b)we computed a 95% equal tail credible interval and HPD. Below value shows the difference of values for both method.

```
[1] 1
```

```
95% Equal tailed credible interval for posterior:
Lower Interval:  0.3907731
UpperInterval:  0.1591066
```

```
95% Equal tailed credible interval for posterior:
Lower Interval:  0.1728546
UpperInterval:  0.413131
```




Equal tail interval cuts the interval from side of the distribution while HPD considers the highest density from top to the bottom and cuts the distribution horizontally

## 3 3. Von Mises distribution.

a.) Posterior distribution plot -



$$Likelihood : P(y|\mu,k) \sim \frac{exp(k*cos(y-\mu))}{2*\pi*I_0(k)}, where(-\pi <= y <= \pi), k > 0,$$

$$Known : \mu = 2.39, \lambda = 1,$$

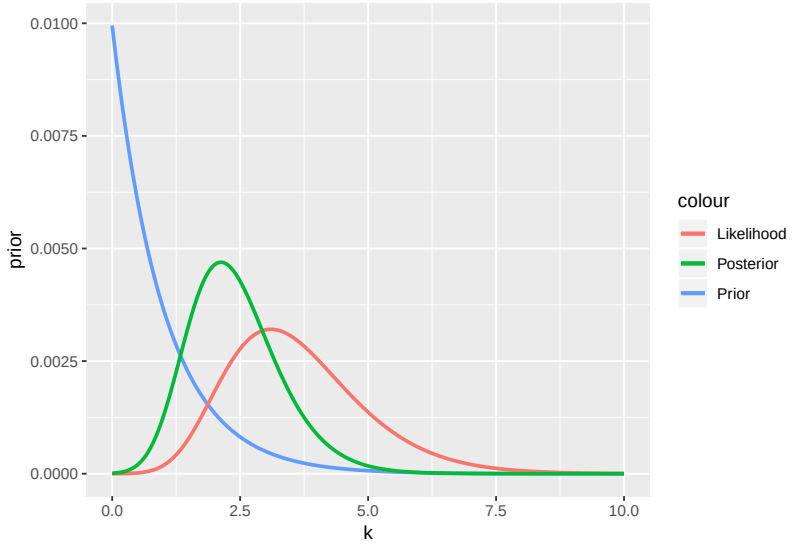
$$Data : y = (-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02),$$

$$Prior : P(k) = \lambda * exp(-\lambda * k) = exp(-k),$$

$$Posterior : posterior \propto likelihood * prior$$

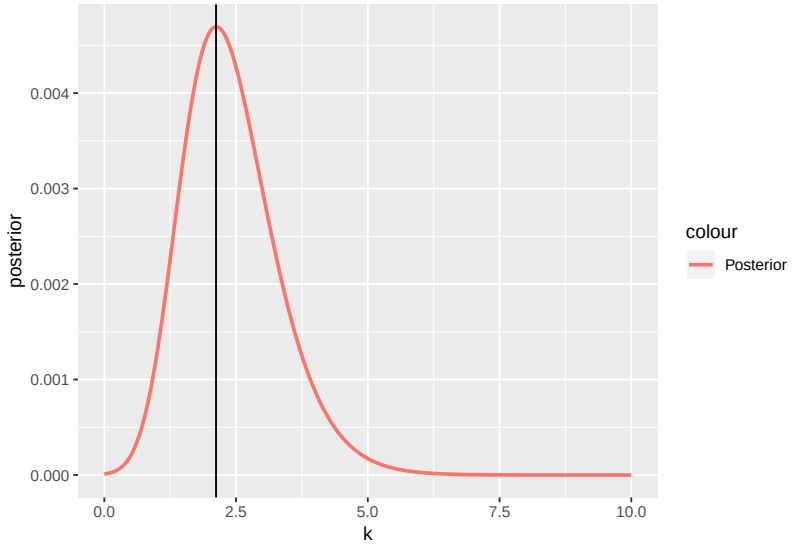
$$P(k|y,\mu) \propto P(y|\mu,k) * P(k)$$

$$P(k|y,\mu) \propto \frac{exp(k*cos(y-\mu))}{2*\pi*I_0(k)} * exp(-k)$$




We normalized the likelihood, prior and the posterior so that we get a better understanding of the calculation by plottiing them on the same plot. I chose the range 0-10 for possible k values. From the posterior, we can see that the most probable value for k is somewhere between (2 and 2.5).

b. ) Approximate Mode of the posterior distribution -




```
The approximate mode for the distribution is :  2.12
```



# 4 Appendix

```
knitr::opts_chunk$set(
    echo = FALSE,
    eval=TRUE,
    message = FALSE,
    warning = FALSE,
    comment = NA
)

library(ggplot2)
library(gridExtra)
library(LaplacesDemon)
alpha <- 16
beta <- 8

beta_mean <- alpha/(alpha+beta)
beta_var <- alpha*beta/(((alpha+beta)^2)*(alpha+beta+1))

set.seed(1234)
pos_beta1 <- rbeta(2,alpha,beta,ncp=0)
pos_beta2 <- rbeta(30,alpha,beta,ncp=0)
pos_beta3 <- rbeta(5000,alpha,beta,ncp=0)


ggplot() + stat_density(aes(x=pos_beta1, color="pos_beta1"),geom="line",size=1) +
           stat_density(aes(x=pos_beta2, color="pos_beta2"),geom="line",size=1) +
           stat_density(aes(x=pos_beta3, color="pos_beta3"),geom="line",size=1) +
           geom_vline(xintercept = beta_mean,linetype="dashed") +
           annotate("text", label = paste0("mean = ", round(beta_mean,2)), size = 4, x = beta_mean+0.06
    scale_color_discrete(name="Number of samples",labels=c("n=2","n=30","n=5000")) + xlab("Samples")+ y
    theme_bw()


mn <- c()
s_dev <- c()
j<-1
sq <- seq(2,10000,length=1000)

for (i in sq ) {
    posbeta <- rbeta(i,alpha,beta,ncp=0)
    mn[j] <- mean(posbeta)
    s_dev[j] <- sd(posbeta)
    j<- j+1
}

cat ("\n")
plot1 <- ggplot() + geom_line(aes(x = sq, y = s_dev)) + ggtitle("Standard deviation convergence of Beta
    xlab("Sample Size ") + ylab("Standard deviation") +
    scale_color_manual(name = NULL, values = c("red"),labels = "True std. deviation")

cat ("\n")
plot2 <- ggplot() + geom_line(aes(x = sq, y = mn)) + ggtitle("Convergence of mean of  Beta posterior")
```

```
    geom_point(aes(x = 10000, y = beta_mean, color = "red")) +
    xlab("Sample Size") + ylab("Mean") +
    scale_color_manual(name = NULL, values = c("red"),labels = "True Mean")

grid.arrange(plot1,plot2, nrow=2)
set.seed(12)
pos_beta <- rbeta(10000,alpha,beta)
pos_beta_l <- sum(pos_beta<0.4)

computed_val <- pos_prob <- pos_beta_l/10000

true_val <- pbeta(0.4,alpha,beta)

cat("Computed value of CDF :",computed_val)
cat("\nTrue value of CDF     :",round(true_val,5))

beta_mean_log <- log(beta_mean/(1-(beta_mean)))
pos_beta_log <- log(pos_beta/(1-pos_beta))

ggplot() + geom_histogram(aes(x = pos_beta_log), color = "black") +
           geom_vline(xintercept = beta_mean_log,linetype="dashed", color="red") +
           annotate("text", label = paste0("mean = ", round(beta_mean_log,2)), size = 4, x = beta_mean_l
    ggtitle("Log odds plot of beta posterior") + xlab("Log odds sample of beta posterior") +  ylab("Dens
set.seed(12345)
mu = 3.5
y = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = 10
tow = sum((log(y) - mu)^2)/n

x = seq(0.1, 5, 0.01)
theo_inv_chi = dinvchisq(x, n, tow)
simu_inv_chi = rinvchisq(10000, n, tow)
ggplot() +
  geom_line(aes(x, theo_inv_chi, col="Theoritical"), size=1) +
  geom_density(aes(simu_inv_chi, col="Simulated"), size=1) +
  ylab("Dennsity") +
  xlab("Inv-Chi-Square")

g_vals = (2*pnorm(sqrt(simu_inv_chi/2))) - 1
ggplot() + geom_density(aes(g_vals)) + ggtitle("Gini Coeffecient of posterior draw")
g_dens = density(g_vals)
data_for_hpd = data.frame(giniVals = g_dens$x, giniDens = g_dens$y)
data_for_hpd$giniDens = data_for_hpd$giniDens/sum(data_for_hpd$giniDens)

data_for_hpd = data_for_hpd[order(data_for_hpd$giniDens, decreasing = TRUE), ]
data_for_hpd$cumPost = cumsum(data_for_hpd$giniDens)
max(data_for_hpd$cumPost)
hpd_vals = which(data_for_hpd$cumPost<0.95)
hpd_cut = length(hpd_vals)

cat("95% Equal tailed credible interval for posterior:\nLower Interval: ",
    data_for_hpd$giniVals[hpd_cut], "\nUpperInterval: ",
    data_for_hpd$giniVals[hpd_cut-1], "\n\n\n")
```



```
quant = quantile(g_vals, c(0.025, 0.975))
cat("95% Equal tailed credible interval for posterior:\nLower Interval: ",
    quant[1], "\nUpperInterval: ",
    quant[2], "\n")

ggplot() + geom_line(aes(data_for_hpd$giniVals, data_for_hpd$giniDens)) +
  geom_hline(aes(yintercept = data_for_hpd$giniDens[hpd_cut], linetype = "HPD"), color = "red", size =
 geom_vline(aes(xintercept = quant[1],linetype = "Equal tail interval"), color = "blue", size = 1) +
  geom_vline(aes(xintercept = quant[2], linetype = "Equal tail interval"), color = "blue", size = 1) +
  xlab("Gini Values") +
  ylab("Density") +
  scale_linetype_manual(name="Linetype", values = c(1,2), guide = guide_legend(override.aes = list(colo
#A3
mu = 2.39
y = c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
prior_a3 = function(k){
  return(exp(-k))
}
likelihood_a3 = function(y, k, mu){
  numer = exp(k*(sum(cos(y - mu))))
  dinom = (2 * pi * besselI(k, 0))^10
  return(numer/dinom)
}

k = seq(0, 10, 0.01)
vals = data.frame(k, prior=prior_a3(k), likelihood=likelihood_a3(y, k, mu))
vals$posterior = vals$prior*vals$likelihood
prior_sum = sum(vals$prior)
likel_sum = sum(vals$likelihood)
postr_sum = sum(vals$posterior)
vals$prior = vals$prior/prior_sum
vals$likelihood = vals$likelihood/likel_sum
vals$posterior = vals$posterior/postr_sum

ggplot(vals) + geom_line(aes(k, prior, col="Prior"), size=1) +
  geom_line(aes(k, likelihood, col="Likelihood"), size=1) +
  geom_line(aes(k, posterior, col="Posterior"), size=1)
max_ind = order(vals$posterior, decreasing = TRUE)[1]
mode_aprox = vals$k[max_ind]
ggplot(vals) + geom_line(aes(k, posterior, col="Posterior"), size=1) +
  geom_vline(xintercept = mode_aprox)
cat("The approximate mode for the distribution is : ", mode_aprox)
```

# Lab2

*Naveen Gabriel(navga709) Sridhar Adhikarla(sriad858)*

*5 May 2019*

## Contents

**1 Linear and polynomial regression** **2**

1 Determining the prior distribution of the model parameters . . . . . 2

2 Simulate from the joint posterior distribution of $\beta_0$ ,$\beta_1$ ,$\beta_2$ and $\sigma^2$ . . . . . 3

3. Locate the time with the highest expected temperature . . . . . 4

4. Mitigate overfitting using prior . . . . . 5

**2. Posterior approximation for classification with logistic regression** **7**

1 Logistic regression . . . . . 7

2. Approximate the posterior distribution of the 8-dim parameter and credible interval . . . . . 7

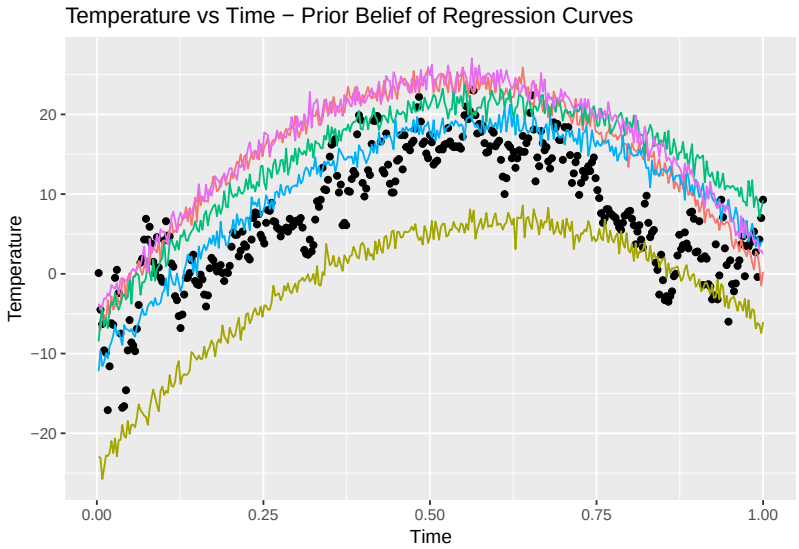
3. Function that simulates from the predictive distribution of the response variable in a logistic regression . . . . . 9

**Appendix** **10**



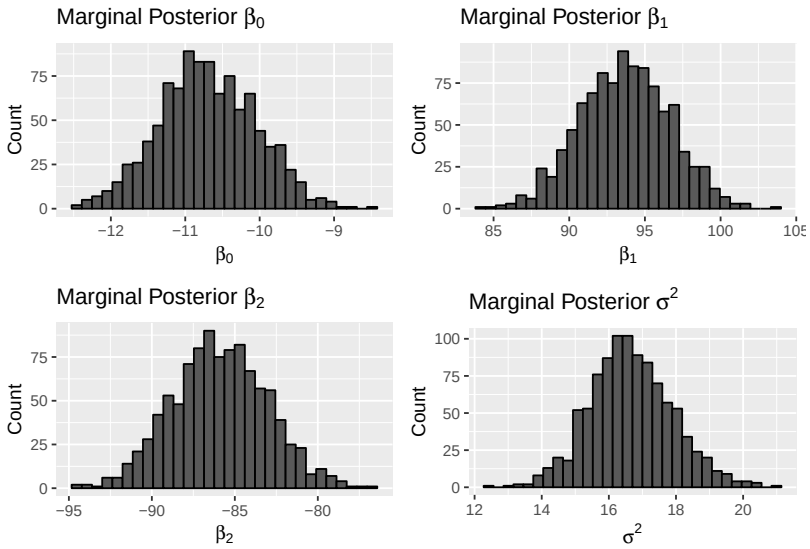
# 1 Linear and polynomial regression

## 1 Determining the prior distribution of the model parameters



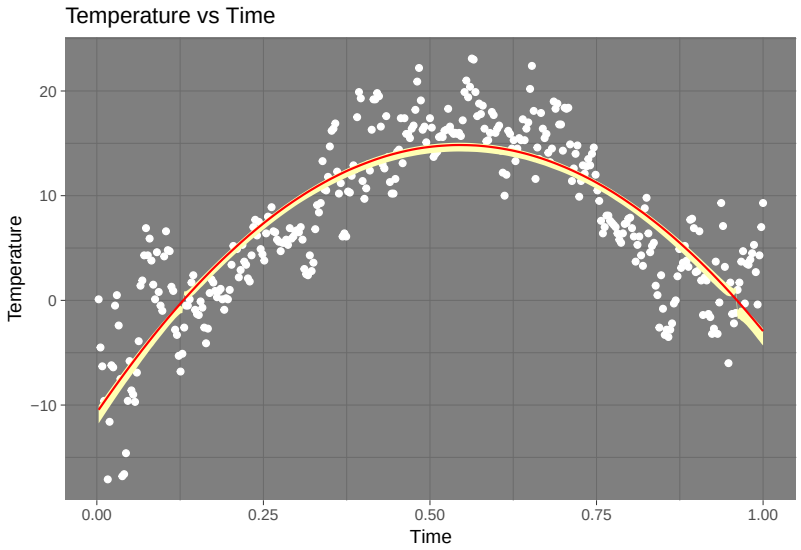

Our prior beliefs on seeing the data does coincides with the group of regression curves which are generated using conjugate prior.

**2 Simulate from the joint posterior distribution of $\beta_0$ ,$\beta_1$ ,$\beta_2$ and $\sigma^2$**




The above plot shows the histogram of marginal posteriors for each parameter, i.e $\beta_0$ ,$\beta_1$ ,$\beta_2$ and $\sigma^2$. Each parameter distribution is normal.



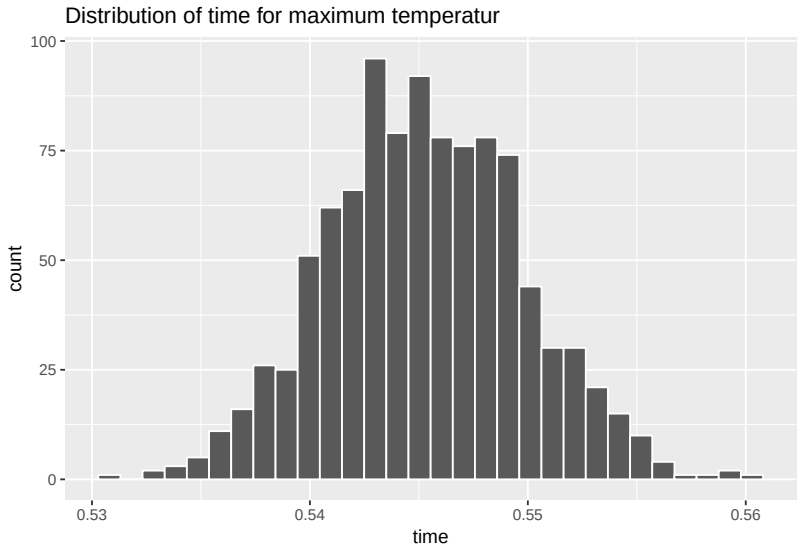



The yellow shows the band with 95% credible interval for f(time). The red curve shows the posterior median of regression curve. It does not include all the point and it should not. The idea is here to find our regression curve which is more tighter and explain the variation of temperature with time.

### 3. Locate the time with the highest expected temperature

Differentiaing the quadratic equation, we get the maximum value of time. We know it is the maximum value because the double differential is negative. We then get the distribution of time where the temperature is maximum from the marginal posterior value of $\beta_0$ and $\beta_2$

$$f(time) = \beta_0 + \beta_1.time + \beta_2.time^2$$
$$f'(time) = 0$$
$$\beta_1 + 2.\beta_2.time = 0$$
$$time = -\frac{\beta 1}{2\beta_2}$$




From the distribution, it seems the time where the temperature is maximum is between 0.54 and 0.55 which rounds off to 197 days approx which comes in June.

## 4. Mitigate overfitting using prior

We can minimize overfitting by changing th variance of our prior

We know :

$$P(\theta|Y) = P(Y|\theta)P(\theta)$$

Assuming zero-mean normally distributed prior on each $\beta_i$ value, all with identical variance $\tau$. Likelyhood Prior is given as :

$$P(Y|\theta) = \prod_{i=1}^{n} \frac{1}{\sigma.\sqrt{2.\pi}}.e^{-\frac{y_i-(\beta_0+\beta_1 x_i+\beta_2 x_i..\beta_p x_p)}{2\sigma^2}}$$

$$P(\theta) = \prod_{j=1}^{p} \frac{1}{\tau.\sqrt{2.\pi}}.e^{-\frac{\beta_j^2}{2\tau^2}}$$

The posterior log can then be calculated as :

$$P(\theta|Y) = log(\prod_{i=1}^{n} \frac{1}{\sigma.\sqrt{2.\pi}}.e^{-\frac{y_i-(\beta_0+\beta_1 x_i+\beta_2 x_i..\beta_p x_p)}{2\sigma^2}}) + log(\prod_{j=1}^{p} \frac{1}{\tau.\sqrt{2.\pi}}.e^{-\frac{\beta_j^2}{2\tau^2}})$$

$$P(\theta|Y) = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i..\beta_p x_p))^2 + \frac{1}{\tau^2}\sum_{j=1}^{p}\beta_j^2$$

$$P(\theta|Y) = \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i..\beta_p x_p))^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$



Here $1/\tau^2$ is $\lambda$ which is our regularization term. We can adjust the amount of regularization we want by changing ??. Equivalently, we can adjust how much we want to weight the priors carry on the coefficients (??). If we have a very small variance (large ??) then the coefficients will be very close to 0; if we have a large variance (small ??) then the coefficients will not be affected much (similar to as if we didn't have any regularization). Which means by increasing the value of lambda our $\Omega_0$ will have low variance

**Source :** http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

## 2. Posterior approximation for classification with logistic regression

### 1 Logistic regression

```
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
```

### 2. Approximate the posterior distribution of the 8-dim parameter and credible interval

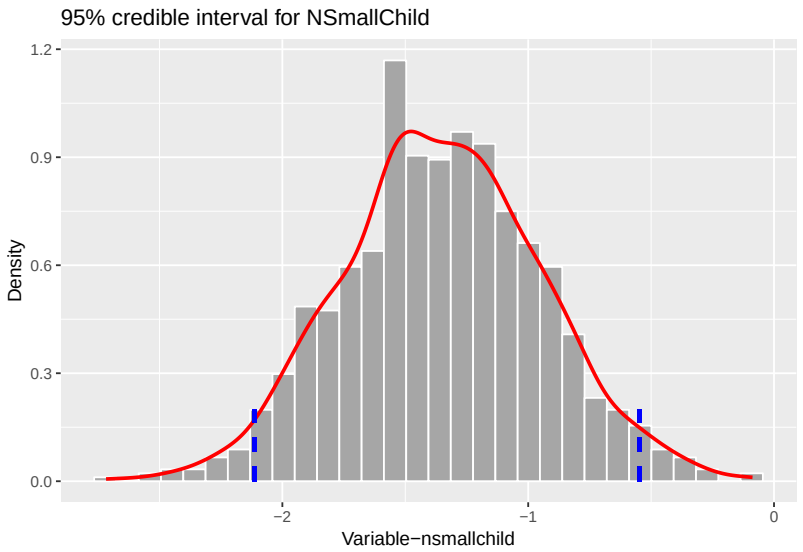
Table 1: Coeffecient value of variables

| | Coeffecient |
|---|---|
| Constant | 0.6267288 |
| HusbandInc | -0.0197911 |
| EducYears | 0.1802190 |
| ExpYears | 0.1675667 |
| ExpYears2 | -0.1445967 |
| Age | -0.0820656 |
| NSmallChild | -1.3591332 |
| NBigChild | -0.0246835 |

```
Hessian matrix:

             [,1]          [,2]          [,3]          [,4]          [,5]
```

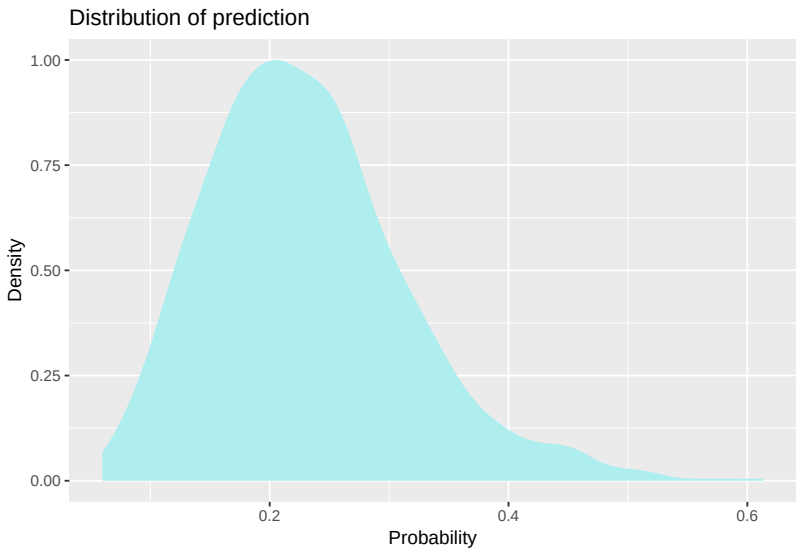


```
[1,]  2.266022568  3.338861e-03 -6.545121e-02 -1.179140e-02  0.0457807243
[2,]  0.003338861  2.528045e-04 -5.610225e-04 -3.125413e-05  0.0001414915
[3,] -0.065451206 -5.610225e-04  6.218199e-03 -3.558209e-04  0.0018962893
[4,] -0.011791404 -3.125413e-05 -3.558209e-04  4.351716e-03 -0.0142490853
[5,]  0.045780724  1.414915e-04  1.896289e-03 -1.424909e-02  0.0555786706
[6,] -0.030293450 -3.588562e-05 -3.240448e-06 -1.340888e-04 -0.0003299398
[7,] -0.188748354  5.066847e-04 -6.134564e-03 -1.468951e-03  0.0032082535
[8,] -0.098023929 -1.444223e-04  1.752732e-03  5.437105e-04  0.0005120144
             [,6]          [,7]          [,8]
[1,] -3.029345e-02 -0.1887483542 -0.0980239285
[2,] -3.588562e-05  0.0005066847 -0.0001444223
[3,] -3.240448e-06 -0.0061345645  0.0017527317
[4,] -1.340888e-04 -0.0014689508  0.0005437105
[5,] -3.299398e-04  0.0032082535  0.0005120144
[6,]  7.184611e-04  0.0051841611  0.0010952903
[7,]  5.184161e-03  0.1512621814  0.0067688739
[8,]  1.095290e-03  0.0067688739  0.0199722657
```




**Would you say that this feature is an important determinant of the probability that a women works?** By looking at the distribution of nsmallChild parameter, 95% credible interval values lies between -2.1 and -0.5 peaking around -1 which means it negatively effect the outcome. So having a child would mean that the women does not work. For greater value of nsmallchild,the value becomes more negative and vice a versa. With this thinking we believe that this variable does effect the outcome.

**3. Function that simulates from the predictive distribution of the response variable in a logistic regression**




From the above distribution most the probability value is below 0.5 and peaks at 0.2 which means 0 has high chances to be the value for classification which means that the women does not work for the given set of parameters.



# Appendix

```
knitr::opts_chunk$set(echo = FALSE,
                      eval = TRUE,
                      warning = FALSE,
                      comment = NA,
                      message=FALSE)
library(ggplot2)
library(mvtnorm)
library(gridExtra)

set.seed(123456789)
templink <- read.delim("TempLinkoping.txt")
n <- nrow(templink)

templink$time_2 <- templink$time^2
templink$firstvar <- 1
templink <- templink[,c(4,1,3,2)]

templink <- as.matrix(templink)
templink_n <- templink
#Setting up parameters for our prior
s_sq <- 1
omga0 <- 0.01*diag(3)
mu0 <- c(-10,100,-100)
v0 <- 4

invchisq <- function(v,sq) {
  sg_sq <- (v*sq)/rchisq(1,v)
  return(sg_sq)
}

#Drawing random variable from chi square
##Conjugate prior
draws <- 8
sg_sq <- invchisq(v0,s_sq)
var_cov <- sg_sq*solve(omga0)

for (i in 1:draws) {
theta <- rmvnorm(1,mu0,var_cov)
y_pred <- templink[,-which(colnames(templink) == "temp")]%*%t(theta) + rnorm(n,0,sg_sq)

templink_n <- cbind(templink_n,y_pred)
}

#drawing theta from Normal distribution
templink_n <- as.data.frame(templink_n)

ggplot(templink_n,aes(time,temp)) + geom_point() +
  geom_line(aes(y=V5,col="V5")) +
  geom_line(aes(y=V6,col="V6")) +
  geom_line(aes(y=V7,col="V7")) +
  geom_line(aes(y=V8,col="V8")) +
```

```
  geom_line(aes(y=V9,col="V9")) +
  ggtitle("Temperature vs Time - Prior Belief of Regression Curves") + xlab("Time") + ylab("Temperature'
  theme(legend.position = "none")



model = lm(temp ~ 1+time+time_2, data = templink_n)
betahat = model$coefficients

mun <- solve(t(templink[,c(1:3)])%*%templink[,c(1:3)] + omga0) %*% (t(templink[,c(1:3)])%*%templink[,c(:
omegan <- t(templink[,c(1:3)])%*%templink[,c(1:3)] + omga0
vn <- v0+n
vn_sigsq <- v0*s_sq + t(templink[,4])%*%templink[,4] + t(mu0)%*%omga0%*%mu0 - t(mun)%*%omegan%*%mun
sg_sq_n  <- vn_sigsq/vn

draws_pos <- 1000

marginal_pos <- matrix(ncol=4,nrow = draws_pos)
colnames(marginal_pos)<- c("beta_0","beta_1","beta_2","sig")

for( i in 1:draws_pos) {
  sg_sq <- invchisq(vn,sg_sq_n)
  var_cov <- as.numeric(sg_sq)*solve(omegan)
  theta <- rmvnorm(1,mun,var_cov)
  marginal_pos[i,] <- c(theta,sg_sq)
}

marginal_pos <- as.data.frame(marginal_pos)

p1 <- ggplot(marginal_pos) + geom_histogram(aes(beta_0),color="black") + xlab (expression(beta[0])) + y:
                      ggtitle(expression(paste("Marginal Posterior ",beta[0])))
p2 <- ggplot(marginal_pos) + geom_histogram(aes(beta_1),color="black") + xlab (expression(beta[1])) + y:
                      ggtitle(expression(paste("Marginal Posterior ",beta[1])))
p3 <- ggplot(marginal_pos) + geom_histogram(aes(beta_2),color="black") + xlab (expression(beta[2])) + y:
                      ggtitle(expression(paste("Marginal Posterior ",beta[2])))
p4 <- ggplot(marginal_pos) + geom_histogram(aes(sig),color="black") + xlab (expression(sigma^2)) + ylab
                      ggtitle(expression(paste("Marginal Posterior ",sigma^2)))

grid.arrange(p1,p2,p3,p4, nrow=2)

pose_table <- matrix(ncol=draws_pos,nrow=n)

for(i in 1:draws_pos) {
  beta0 <- marginal_pos[i,]$beta_0
  beta1 <- marginal_pos[i,]$beta_1
  beta2 <- marginal_pos[i,]$beta_2

  pose_table[,i] <-  beta0 + beta1*templink_n$time + beta2*templink_n$time_2
}


low_high <- function(x) {
  x <- x[order(-x)]
```



```
  y <- cumsum(x)/sum(x)

  ind_high <- max(which(y<=0.25))
  ind_low <- min(which(y>=0.975))

  high_time <- x[ind_high]
  low_time <- x[ind_low]

  return(c(high_time,low_time))
}


templink_n$pos <- apply(pose_table,1,median)

low_high_val <- apply(pose_table,1,FUN=function(x) low_high(x))

templink_n$low_pos <- low_high_val[2,]
templink_n$high_pos <- low_high_val[1,]

#With posterior median and upper and lower 2.5% confidence
ggplot(templink_n,aes(time,temp)) + geom_point(color="white") +
  geom_ribbon(aes(ymin=low_pos, ymax=high_pos),fill = "#ffffb3") +
  geom_line(aes(y=pos),color="red",size=0.6) + theme_dark() +
  ggtitle("Temperature vs Time") + xlab("Time") + ylab("Temperature")

marginal_pos$time <- apply(marginal_pos,1,FUN=function(x) -x[2]/(2*x[3]))

ggplot(marginal_pos,aes(time)) + geom_histogram(color="white") +
  ggtitle("Distribution of time for maximum temperatur ")
womenwork <- read.table("womenWork.dat",header = TRUE)

model <- glm(Work~0+.,data = womenwork,family = "binomial")

summary(model)
x <- as.matrix(womenwork[,-1])
y <- womenwork[,1]
covnames <- colnames(x)
tau <- 10


LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
```

```
  return(logLik + logPrior)
}

initVal <- as.vector(rep(0,dim(x)[2]))
# Setting up the prior
mu <- as.vector(rep(0,dim(x)[2])) # Prior mean vector
Sigma <- tau^2*diag(dim(x)[2]);

OptimResults<-optim(initVal,LogPostLogistic,gr=NULL,y,x,mu,
                    Sigma,method=c("BFGS"),control=list(fnscale=-1),
                    hessian=TRUE)

postMode <- OptimResults$par
names(postMode) <- covnames
postCov <- -solve(OptimResults$hessian) # Posterior covariance matrix is -inv(Hessian)
approxPostStd <- sqrt(diag(postCov)) # Computing approximate standard deviations.
names(approxPostStd) <- covnames # Naming the coefficient by covariates

postmode <- as.data.frame(postMode)
colnames(postmode) <- "Coeffecient"
knitr::kable(data.frame(postmode),caption="Coeffecient value of variables")
cat("Hessian matrix:\n")
postCov
mu_nsmall <- postMode["NSmallChild"]
sd_n_small <- approxPostStd["NSmallChild"]

dist <- as.data.frame(rnorm(1000,mu_nsmall,sd_n_small))
colnames(dist) <- "var"

intv <- quantile(dist$var, probs = c(0.025, 0.975))

ggplot(dist, aes(x=var)) + geom_histogram(aes(y = ..density..), color= "white", fill="#a6a6a6") +
  stat_density(geom="line", color="red", size=1) +
  geom_segment(aes(x = intv[1], y = 0, xend = intv[1], yend = 0.20),linetype="dashed",color="blue", size
  geom_segment(aes(x = intv[2], y = 0, xend = intv[2], yend = 0.20),linetype="dashed",color="blue", size
  ggtitle("95% credible interval for NSmallChild") + xlab("Variable-nsmallchild") + ylab("Density")


predict_dist <- function(pred_data, optimres,sampl) {

  var_cov <- rmvnorm(1000,optimres$par,-solve(optimres$hessian))
  p <- c()

  for (i in 1:sampl) {
    p[i] <- (exp(pred_data %*% var_cov[i, ]))/(1 + (exp(pred_data %*% var_cov[i, ])))
  }
  return(p)

}


pred_data <- c(1, 10, 8, 10, (10/10)^2, 40, 1,1)
```



```
predic <- predict_dist(pred_data,OptimResults,1000)

ggplot() + stat_density(aes(x=predic, y=..scaled..),fill="#AFEEEE") + xlab("Probability") +
           ylab("Density") + ggtitle("Distribution of prediction")
```

# Lab-3

*Naveen Gabriel ,Sridhar Adhikarla*

*2019-05-19*

## Contents

**1 Normal model, mixture of normal model with semi-conjugate prior. 2**
1.1 Normal model. . . . 2
1.2 Mixture Model . . . 9
1.2.1 Convergence of sampler . . . 9
1.2.2 Parameter Trajectories . . . 10
1.3 Graphical Comparison . . . 11

**2 Poisson regression - the MCMC way. 11**
2.1 A -> Fitting a regression model on the data (GLM) . . . 11
2.2 B -> Bayesian analysis of the Poisson regression . . . 11
2.3 C -> Simulate Posterior using Metropolis algorithm . . . 12

**3 Appendix 16**



# 1 Normal model, mixture of normal model with semi-conjugate prior.

## 1.1 Normal model.

Assuming the daily precipitaion follows normal distribution whre both $\mu$ and $\sigma^2$ are unknown.

$$y_1, y_2...y_n \sim \mathcal{N}(\mu, \sigma^2)$$

Assuming our semi-conjugate priors $\mu$ and $\sigma$ follows below distribution :

$$\mu \sim \mathcal{N}(\mu_0, tau_0^2)$$
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

So our full conditional posterior is :

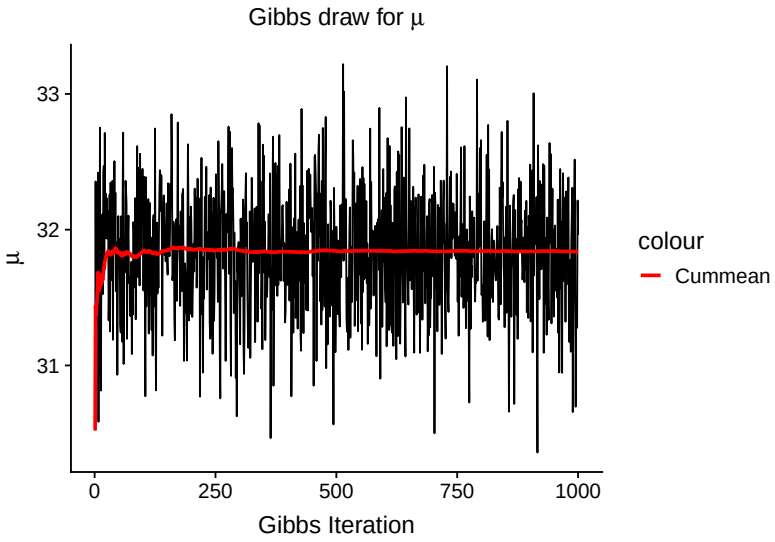
$$\mu|\sigma^2, x \sim \mathcal{N}(\mu_n, tau_n^2)$$
$$\sigma^2|\mu, x \sim Inv - \chi^2(v_n, \frac{v_o\sigma^2 + \sum_{i=0}^{n}(x_i - \mu)^2}{n + v_0})$$

Where $v_n$ and $tau_n^2$ values are

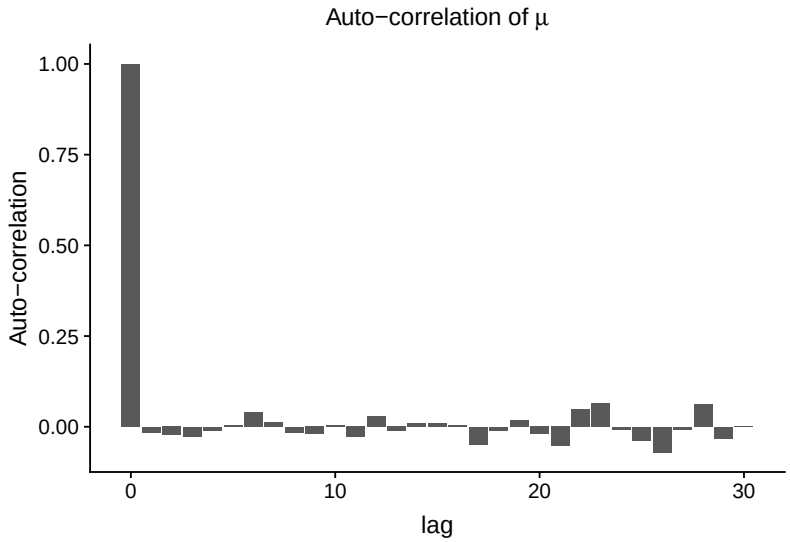
$$\tau_n^2 = \frac{\sigma^2\tau_0^2}{n\tau_0^2 + \sigma^2}$$
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

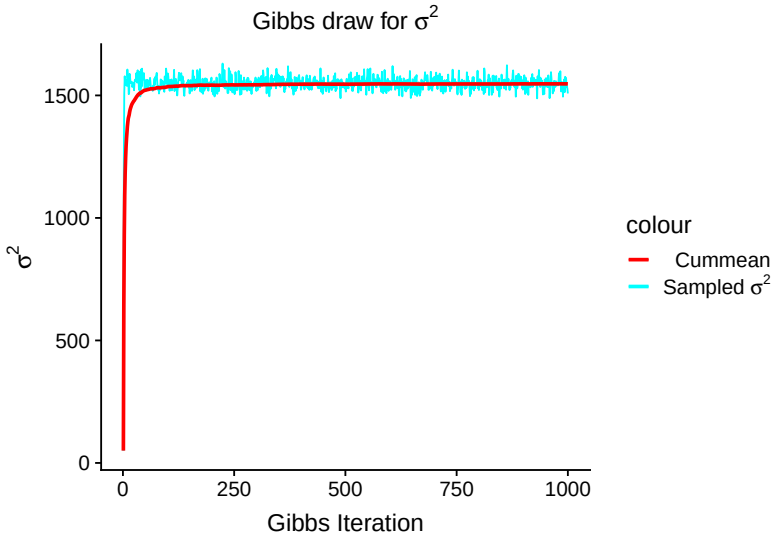
Where w is :

$$w = \frac{n/\sigma^2}{n/\sigma^2 + 1/\tau_0^2}$$

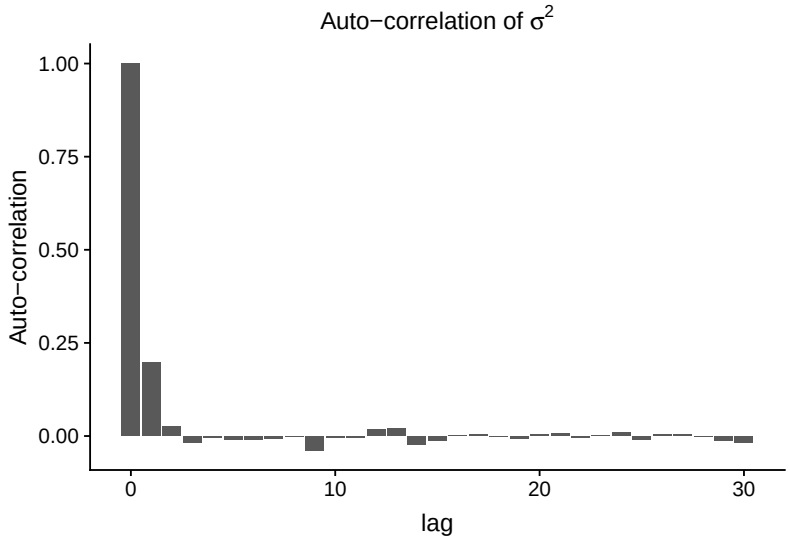

Gibbs draw for μ
33
32
31
μ
0
250
500
750
1000
Gibbs Iteration
colour
Cummean

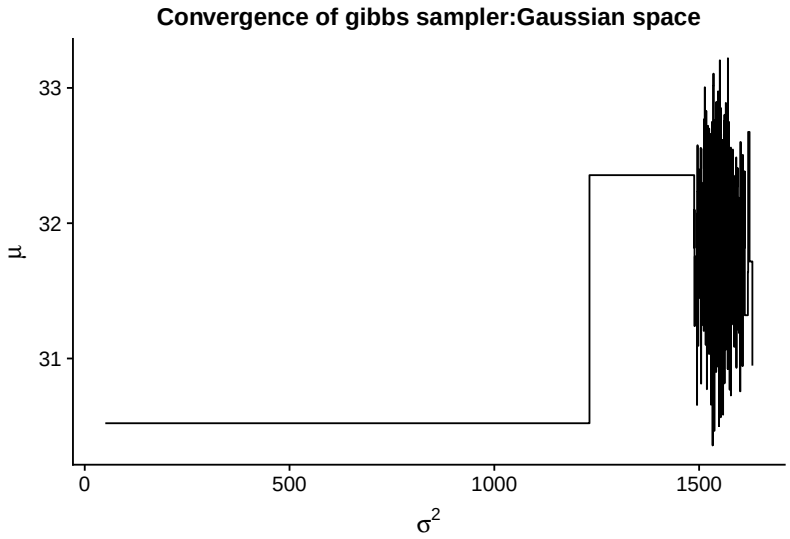



Auto−correlation of μ
1.00
0.75
0.50
0.25
0.00
Auto−correlation
0
10
20
30
lag


Gibbs draw for σ2
1500
1000
500
0
σ2
0
250
500
750
1000
Gibbs Iteration
colour
Cummean
Sampled σ2



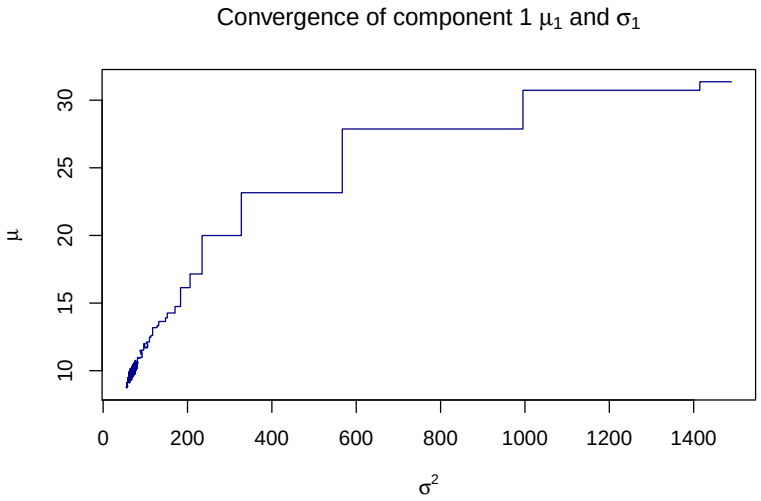

Auto−correlation of σ2
1.00
0.75
0.50
0.25
0.00
Auto−correlation
0
10
20
30
lag

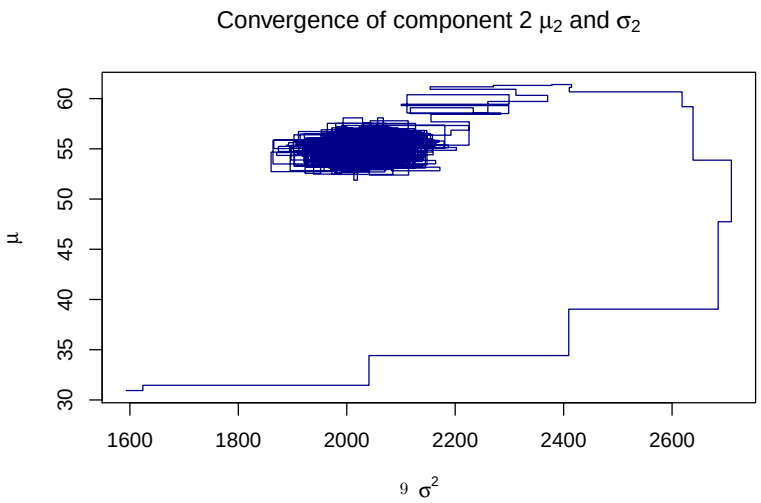



From the above plots, it is clear that using gibbs sampler converges to stable mu and sigma value

## 1.2 Mixture Model

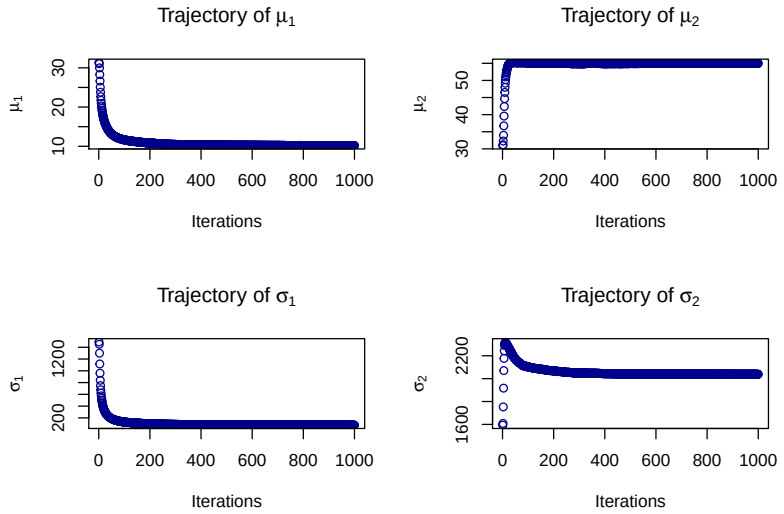
### 1.2.1 Convergence of sampler



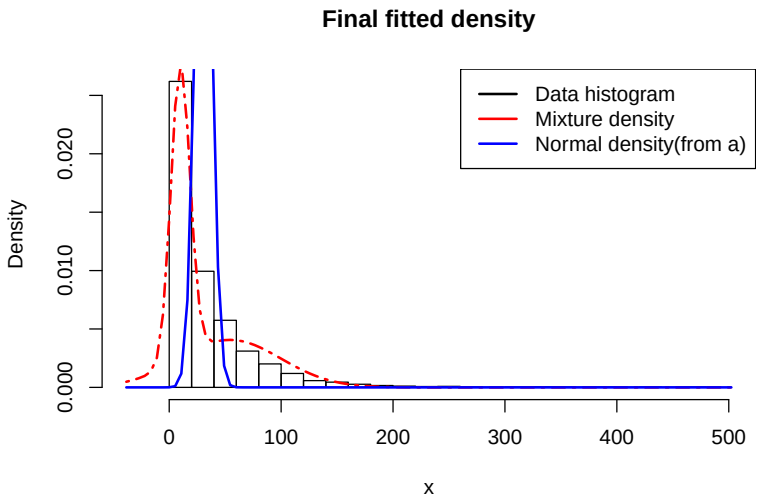



### 1.2.2 Parameter Trajectories




When we consider 2 component mixture of normel model for iid distribution of precipitation, the convergance of sampler and component parameters are visualized using the plots above.

### 1.3 Graphical Comparison




For plotting the normal density from part a, we have considered the mode of sampled values from mu and sigma and using that the normal distribution was plotted for the range of values.

## 2 Poisson regression - the MCMC way.

### 2.1 A -> Fitting a regression model on the data (GLM)

```
## MinBidShare       Const      Sealed    VerifyID     MajBlem     LogBook
## -1.89409664  1.07244206  0.44384257 -0.39451647 -0.22087119 -0.12067761
##     LargNeg     Minblem PowerSeller
##  0.07067246 -0.05219829 -0.02054076
```

We have sorted the covariates according to their absolute significance in the output above. Some of the most significant covariates were - "MinBidShare", "Sealed", "VarifyID", "LogBook".

### 2.2 B -> Bayesian analysis of the Poisson regression

```
## MinBidShare       Const      Sealed    VerifyID     MajBlem     LogBook
## -1.89196574  1.06984694  0.44356304 -0.39301311 -0.22122262 -0.12021722
##     LargNeg     Minblem PowerSeller
##  0.07072379 -0.05248375 -0.02051922
```



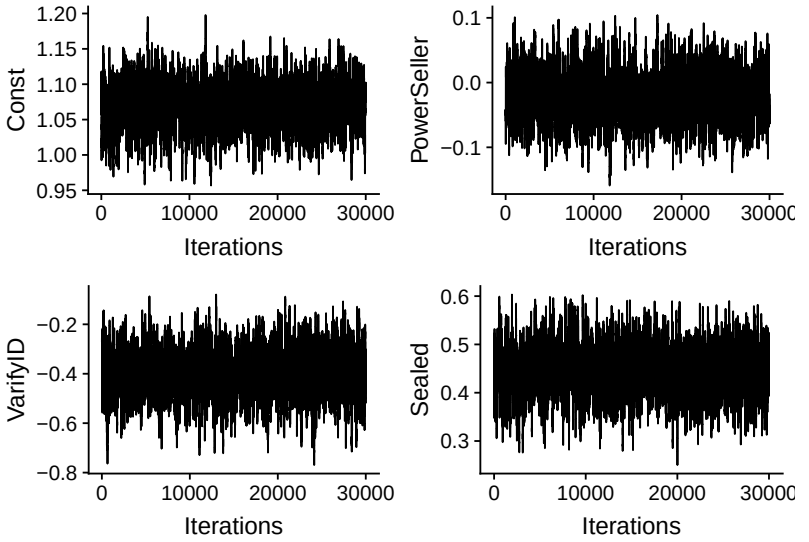
Using numerical optimization(optim function in R) for approximating the posterior distribution of beta as a multivariate normal, we get similar values for the covariates. The order of significance remains the same as we got in the previous question. The covariates "MinBidShare", "Sealed", "VarifyID", "LogBook" are still significant.

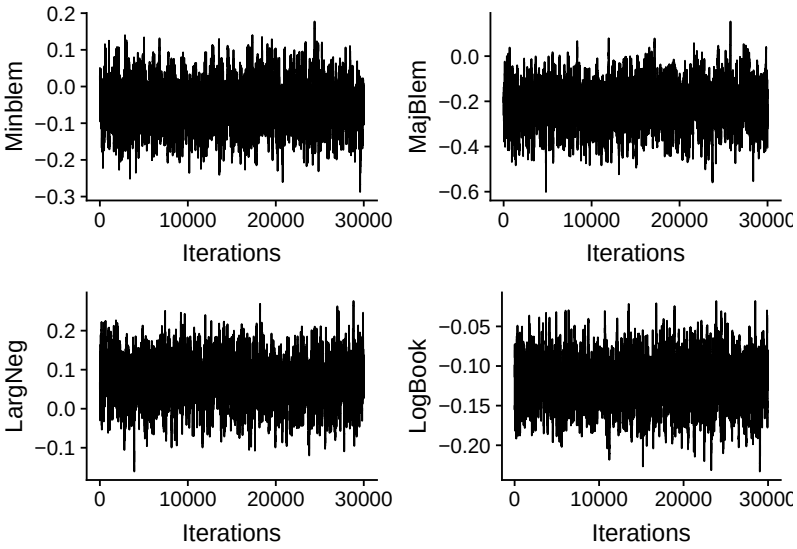
We implemented the logPostProp function for this question, which calculates the sum of log prior and the log likelihood given the initial beta values and the data.

Likelihood values can get very small because we multiply so many small values, and there is a possible loss of numerical precision. This is the reason we use the log of the posterior to prevent this problem.

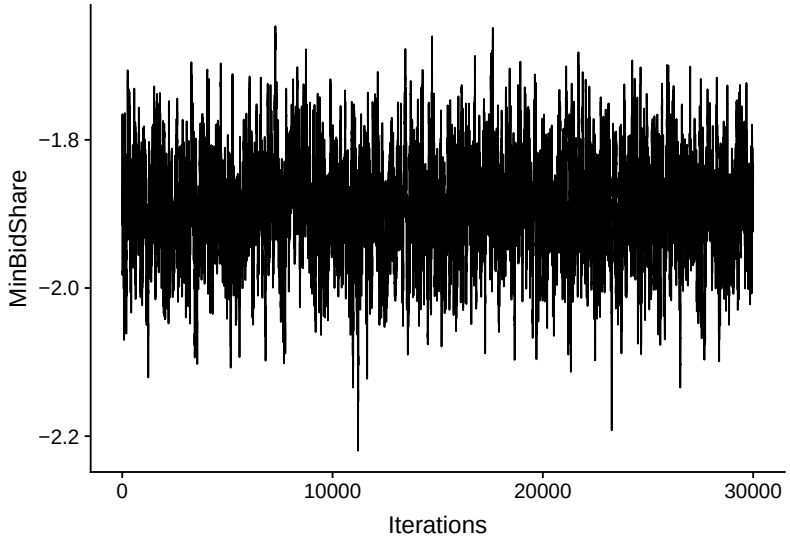
### 2.3 C -> Simulate Posterior using Metropolis algorithm

```
## Acceptance Rate for Metropolis : 0.4374848
```








We implemented a universal function for Metropolis haistings simulation algorithm. This function takes in as arguments the logPosterior function, the initial parameters for the posterior, and a parameter to control the variability of the simulations being generated. Along with this you can provide any number of additional arguments required for your logPosterior function. In our scenario we are sending in the data X, Y as the additional arguments for the function.

We are making 30,000 draws from the posterior, and we add in the additional burn-in period to this as 10 percent of the number of draws we are makin. So Total draws we are making is 33,000 off which we discard the first 3000 draws.

The parameter "c" that controls the variance of the simulation, was adjusted so that we get an acceptance rate close to 40%. We are getting an acceptance rate of : 0.4351212 for this chain, that is 43% accept rate, which is reasonable.

From the above trace plots for the parameters it looks like the Markov chain has reached the stationary distribution.

```
## Effective sample size for Const  :  882.8752
## Effective sample size for PowerSeller  :  874.1789
## Effective sample size for VerifyID  :  862.3082
## Effective sample size for Sealed  :  962.5294
## Effective sample size for Minblem  :  921.9693
## Effective sample size for MajBlem  :  909.1043
## Effective sample size for LargNeg  :  935.1068
## Effective sample size for LogBook  :  899.4706
## Effective sample size for MinBidShare  :  894.6875
```

On further analysis for the convergence of the markov chain, we can see that the effective sample size is also good, approx 850 for all the parameters. So we can get a lot of information from this posterior to make

interval estimates about the parameter values.

```
## [1] "Mean parameter values from the simulated posterior:"
```

```
##         Const PowerSeller   VerifyID    Sealed    Minblem    MajBlem
## [1,] 1.068269 -0.02048009 -0.3990935 0.4457779 -0.0521104 -0.2204971
##         LargNeg    LogBook MinBidShare
## [1,] 0.07148741 -0.1206097   -1.890744
```

The parameter values are similar to the ones we got for the first two questions



# 3 Appendix

```
knitr::opts_chunk$set(echo = FALSE,warning=NA,eval=TRUE,message=FALSE)

library(ggplot2)
library(gridExtra)
library(dplyr)
#to calculate inverse chi sqaure
invchisq <- function(v,sq) {
  sg_sq <- (v*sq)/rchisq(1,v)
  return(sg_sq)
}
rainfall <- read.table("rainfall.txt")
colnames(rainfall) <- "Precipitation"

n <- nrow(rainfall)

#Initial parameters
mu0 <- 30
sg0_sq <- 50
tau0_sq <- 1
nDraws = 1000  #number of draws for parameters
v0 <- 1500


gibbs_bayes <- function(x,mu,sg_sq,tau0_sq,v0)
{ mn <- mean(x)
  n <- length(x)
  vn <- v0 +n
  gibbsDraws <- matrix(0,nDraws,2)
  gibbsDraws[1,] <- c(mu,sg_sq)
  for (i in 2:nDraws) {

    numertr <- (n/sg_sq)
    denomtr <- (n/sg_sq) + (1/tau0_sq)
    w <- (numertr)/(denomtr)
    mu_n <- w*mn + (1-w)*mu0

    taun_sq <-  (sg_sq*tau0_sq)/((n*tau0_sq)+ sg_sq)
    mu <- rnorm(1,mu_n,sqrt(taun_sq))

    scn_arg <- ((v0*sg_sq)+sum((x-mu)^2))/vn
    sg_sq <- invchisq(vn,scn_arg)
    gibbsDraws[i,] <- c(mu,sg_sq)

  }

  return(gibbsDraws)

}

mu <- rnorm(1,mu0,sqrt(tau0_sq))
sg_sq <- invchisq(v0,sg0_sq)
```

```r
data <- gibbs_bayes(rainfall$Precipitation,mu,sg_sq,tau0_sq,v0)
data <- as.data.frame(data)
colnames(data) <- c("mu","sig")
data$cummean_mu <- cummean(data$mu)
data$cummean_sig <- cummean(data$sig)

#calculating autocorrelation

mu_ac <- acf(data$mu,plot=FALSE,type="correlation")
sig_ac <- acf(data$sig,plot=FALSE,type="correlation")
autocor_data <- data.frame(mu_ac$acf,sig_ac$acf,lag=mu_ac$lag)
colnames(autocor_data) <- c("mu_ac","sig_ac","lag")

    ggplot(data, aes(x = 1:nDraws,y =mu, mucolor = "mu")) + geom_line() + xlab("Gibbs Iteration") +
    ylab(expression(mu)) + geom_line(aes(x = 1:nDraws,y =cummean_mu,color = "cummean_mu"), size = 1) +
    ggtitle(expression(paste("Gibbs draw for ", mu))) +
    scale_color_manual(labels = c("Cummean", expression(paste("Sampled ", mu))),
    values = c("red", "Cyan"))

    ggplot(data = autocor_data, mapping = aes(x = lag, y = mu_ac)) +
    geom_bar(stat = "identity", position = "identity") +
    ggtitle(expression(paste("Auto-correlation of ", mu))) + ylab("Auto-correlation")

    ggplot(data, aes(x = 1:nDraws,y = sig,color = "sig")) + geom_line() + xlab("Gibbs Iteration") +
    ylab(expression(sigma ^ 2)) + geom_line(aes(x = 1:nDraws,y = cummean_sig,
                                        color = "cummean_mu"), size = 1) +
    ggtitle(expression(paste("Gibbs draw for ", sigma^2))) +
    scale_color_manual(labels = c("Cummean", expression(paste("Sampled ", sigma ^
    2))),values = c("red", "cyan"))

    ggplot(data = autocor_data, mapping = aes(x = lag, y = sig_ac)) +
    geom_bar(stat = "identity", position = "identity") +
    ggtitle(expression(paste("Auto-correlation of ", sigma ^
    2))) +
    ylab("Auto-correlation")


ggplot(data,aes(x=sig,y=mu)) + geom_step() + xlab(expression(sigma^2)) + ylab(expression(mu)) + ggtitle
#Code credits :  Estimating a simple mixture of normals Author: Mattias Villani
#this uses semi conjugate prior
# Data options
set.seed(12345)
x <- as.matrix(rainfall$Precipitation)

# Model options
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
```



```r
# MCMC options
nIter <- 1000 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
################   END USER INPUT ###############

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

####### Defining a 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)
  }
# Diving every column of piDraws by the sum of the elements in that column.
  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){
    # from all components it return which position has 1
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}

# Initial value for the MCMC
nObs <- length(x)
#S is a matrix which says from which component the
#particular observation is sampled'
# nObs-by-nComp matrix with component allocations.
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(density(x)$y))
#ylim = c(0,2*max(hist(x)$density))
par_mat = matrix(NA, nrow = nIter, ncol = nComp*2)
for (k in 1:nIter){
  alloc <- S2alloc(S) # Just a function that converts between different
                      #representations of the group allocations
  nAlloc <- colSums(S)
  # 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))
  }
   par_mat[k,1:nComp] = mu

  # 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 == j] - mu[j])^2))/(nu0[j] + nAlloc[j]))

  }
  par_mat[k, -c(1:nComp)] = sigma2

  # 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%%1 ==0)){
    effIterCount <- effIterCount + 1

    mixDens <- rep(0,length(xGrid))

    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

  }
```



```
}


plot(x = par_mat[,3], y = par_mat[,1], type = 's',
     main=expression(paste("Convergence of component 1 ",mu[1]," and ", sigma[1])),
     col = "DarkBlue", xlab=expression(paste("",sigma^2)),
     ylab=expression(paste("",mu)))

plot(x = par_mat[,4], y = par_mat[,2], type = 's',
     main=expression(paste("Convergence of component 2 ",mu[2]," and ", sigma[2])),
     col = "DarkBlue",xlab=expression(paste("",sigma^2)),
     ylab=expression(paste("",mu)))

par_mat_cumsum = apply(par_mat, 2, cumsum)
par_mat_cumsum = apply(par_mat_cumsum,2,"/",seq(1:nIter))

par(mfrow =c(2,2))
plot(x = seq(1:nIter) , y = par_mat_cumsum[,1], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",mu[1])), xlab = "Iterations",
     ylab = expression(paste(mu[1])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,2], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",mu[2])), xlab = "Iterations",
     ylab = expression(paste(mu[2])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,3], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",sigma[1])), xlab = "Iterations",
     ylab = expression(paste(sigma[1])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,4], col = "DarkBlue" ,
    main = expression(paste("Trajectory of ",sigma[2])), xlab = "Iterations",
    ylab = expression(paste(sigma[2])))

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
     main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 6, col = "red")
lines(xGrid, dnorm(xGrid, mean = Mode(data$mu), sd = sqrt(Mode(data$sig))),
      type = "l", lwd = 2, col = "blue")
legend("topright", box.lty = 1,
       legend = c("Data histogram","Mixture density","Normal density(from a)"), col=c("black","red","blu
knitr::opts_chunk$set(echo = FALSE, message = FALSE,warning = FALSE)
library(MASS)
library(coda)
library(mvtnorm)
library(LaplacesDemon)
library(cowplot)
#Q1
data = read.table("eBayNumberOfBidderData.dat", header=TRUE)
n = length(data)
```

```
n_features = ncol(data) - 1 # Except y and const
feature_labels = colnames(data[,2:ncol(data)])

y = data$nBids
X = as.matrix(data[,2:ncol(data)])
X_X = t(X)%*%X
glm_model = glm(nBids ~ 0 + ., data = data, family = poisson)

glm_model$coefficients[order(abs(glm_model$coefficients), decreasing = TRUE)]
#summary(glm_model)
# Beta prior (Zellner's g-prior)
mu0 = numeric(n_features)
covar0 = 100 * ginv(X_X)
init_beta = mvrnorm(n=1, mu0, covar0)

# Log Posterior Poisson model
logPostProp <- function(betas, X, y){
  log_prior = dmvnorm(betas, mu0, covar0, log=TRUE)
  lambda = exp(X%*%betas)

  log_lik = sum(dpois(y, lambda, log=TRUE))

  return (log_lik + log_prior)
}

opt_results = optim(init_beta, logPostProp, gr=NULL, X, y,method=c("BFGS"),
                    control=list(fnscale=-1), hessian=TRUE)

# MLE beta
post_mode = opt_results$par
names(post_mode) = names(glm_model$coefficients)
post_cov = -solve(opt_results$hessian)
post_mode[order(abs(post_mode), decreasing = TRUE)]
Sigma = post_cov
c = .3
n_draws = 30000
burn_in = floor(n_draws / 10)
n_draws = n_draws + burn_in

metropolisHastings = function(logPostFunc, theta, c_param, ...){
  theta_draws = matrix(0, n_draws, length(theta))

  # Set initial
  theta_c = mvrnorm(n=1, theta, c_param)
  accpt = 0
  for(i in 1:n_draws){
    # 1: Draw new candidate theta
    theta_p = mvrnorm(n=1, theta_c, c_param)

    # 2: Determine the acceptance probability alpha
    alpha = min(c(1, exp(logPostFunc(theta_p, ...) - logPostFunc(theta_c, ...))))

    # 3: Set new value with prob = alpha
```



```
    if(rbern(n=1, p=alpha)==1){
      theta_c = theta_p
      accpt = accpt + 1
    }
    theta_draws[i,] = theta_c
  }
  cat("Acceptance Rate for Metropolis :", accpt/n_draws, "\n")

  return (theta_draws)
}
init_beta = mvrnorm(n=1, mu0, covar0)
beta_draws = metropolisHastings(logPostProp, init_beta, c*Sigma, X, y)

beta_draws = beta_draws[(burn_in+1):nrow(beta_draws),]
beta_means = t(as.matrix(colMeans(beta_draws)))
colnames(beta_means) = feature_labels

tot = dim(beta_draws)[1]
tp = ggplot() + xlab("Iterations")
plot_grid(tp + geom_line(aes(x = 1:tot, y = beta_draws[,1])) + ylab("Const"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,2])) + ylab("PowerSeller"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,3])) + ylab("VarifyID"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,4])) + ylab("Sealed"))
plot_grid(tp + geom_line(aes(x = 1:tot, y = beta_draws[,5])) + ylab("Minblem"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,6])) + ylab("MajBlem"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,7])) + ylab("LargNeg"),
          tp + geom_line(aes(x = 1:tot, y = beta_draws[,8])) + ylab("LogBook"))
tp + geom_line(aes(x = 1:tot, y = beta_draws[,9])) + ylab("MinBidShare")
for(i in 1:9){
  cat("Effective sample size for", feature_labels[i]," : ",
      effectiveSize(as.mcmc(beta_draws[,i])), "\n")
}
print("Mean parameter values from the simulated posterior:")
beta_means
```

# Lab4

*Naveen Gabriel(navga709) Sridhar Adhikarla(sriad858)*

*2019-05-29*

## Contents

**1 Time series models in Stan** **2**
Question 1A : Function in R that simulates data from the AR(1)-process . . . 2
Question 1B : Stan Model using non-informative prior on X and Y . . . 4
Compiling the model and generating data required . . . 4
Summary from model fit on data X . . . 5
Summary from model fit on data Y . . . 6
Traceplots for the fit models . . . 7
Joint Posterior for mu and phi . . . 8
Question 1C : Stan Model using non-informative prior on campy data . . . 10
Summary from the model fit on campy data . . . 10
Question 1D : Stan Model using Informative prior on X and Y . . . 12
Summary from fitted model . . . 14

**Appendix** **17**

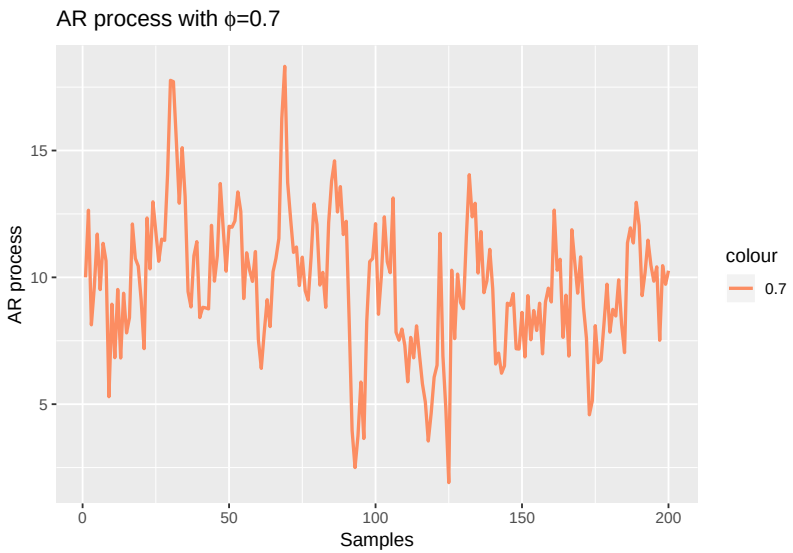


# 1 Time series models in Stan

### Question 1A : Function in R that simulates data from the AR(1)-process

Given AR process :

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




Different $\phi$ has different effect on AR process. For -1 the AR process oscillates the most going over point again and again. As the value of phi increases towards 1 the AR process oscillates decreases and goes through varied different points. We have chosen *phi* as 0.7 for further analysis

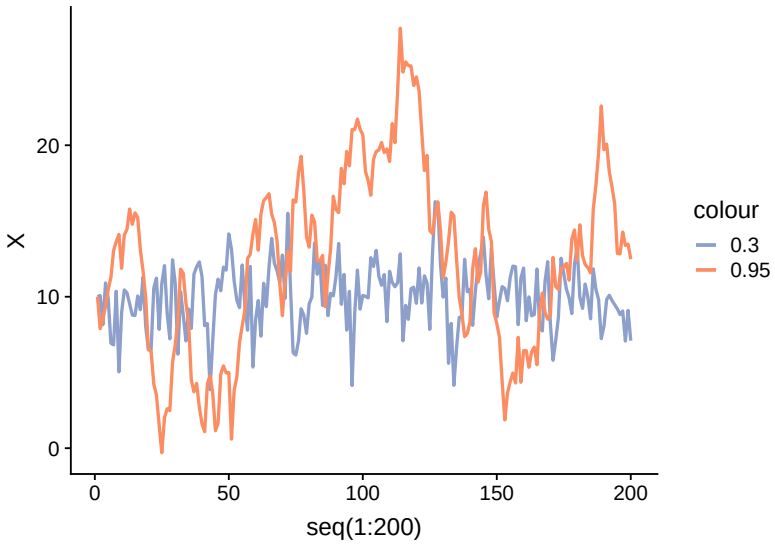


## Question 1B : Stan Model using non-informative prior on X and Y

**Compiling the model and generating data required**

```
#1B
stan_code1 = "data {
  int<lower=0> T;
  vector[T] y;
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  y[2:T] ~ normal(alpha + beta*y[1:(T-1)], sigma);
}
"
#compile model
model <- stan_model(model_code = stan_code1)
```

This is the AR1 model implemented in stan. The priors used for this model are non-informative as it is almost flat in the space of the parameter. All the possible values for the parameter are equally likely as we have no prior knowledge about the data.




This plot shows plots for the data generated for this model, X and Y.

We sample 4 chains of size 1000 from the model for each of the data X and Y. The burn-in preiod is 500, so the first 500 samples will be discarded by the sampler.

According to the AR1 model

beta = phi, and

alpha = mu * (1 - beta)

This is the reason we recompute mu from alpha using the formula mu = alpha / (1 - beta)

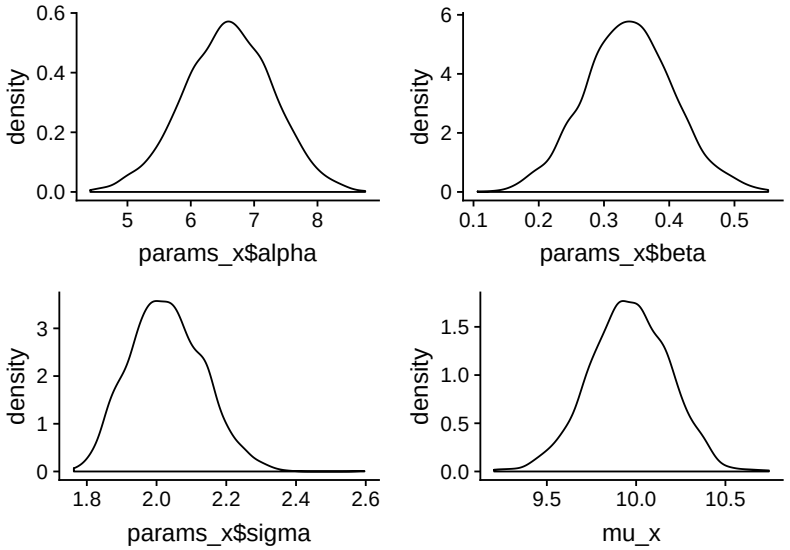
**Summary from model fit on data X**

```
## Mean for mu : 9.962597
```

```
## [1] "95% confidence interval for mu"
```

```
##      2.5%     97.5%
##  9.517727 10.379458
```






```
## [1] "95% confidence intervals for alpha, beta, sigma"
```

```
##         mean   2.5%  97.5% n_eff
## alpha 6.5930 5.1651 7.9403 720.0
## beta  0.3383 0.2023 0.4778 721.1
## sigma 2.0289 1.8453 2.2455 956.0
```

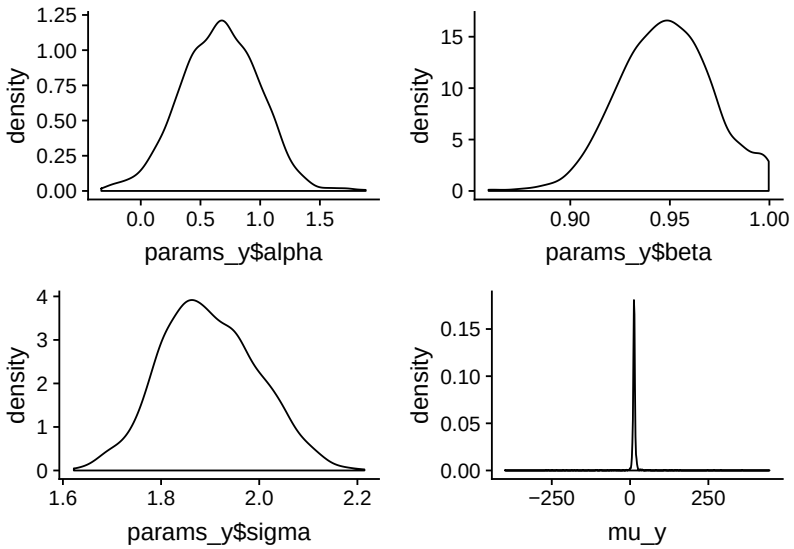
The estimated parameters are close to the original parameters. The estimated value for mu is almost exactly the same as the true value. The true values for all the parameters lie in the 95% credible interval calculated for that parameter.

**Summary from model fit on data Y**

```
## Mean for mu : 12.2213
```

```
## [1] "95% confidence interval for mu"
```

```
##      2.5%     97.5%
##  1.443359 23.446087
```




```
## [1] "95% confidence intervals for alpha, beta, sigma"
```

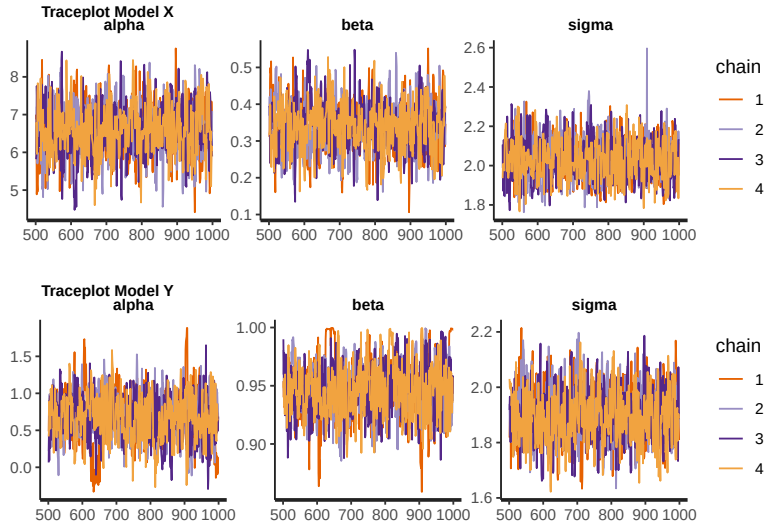
```
##         mean    2.5%  97.5% n_eff
## alpha 0.6639 0.02273 1.2730 456.5
## beta  0.9479 0.90268 0.9966 455.2
## sigma 1.8983 1.71097 2.0944 902.5
```

The estimated parameters are close to the original parameters. The estimated value for mu is a little deviated from the true value. The true values for all the parameters still lie in the 95% credible interval calculated for that parameter.

The effective sample size is large for all the simulated parameters, which is good showing that the parameters are sampled from a stationary distributions.

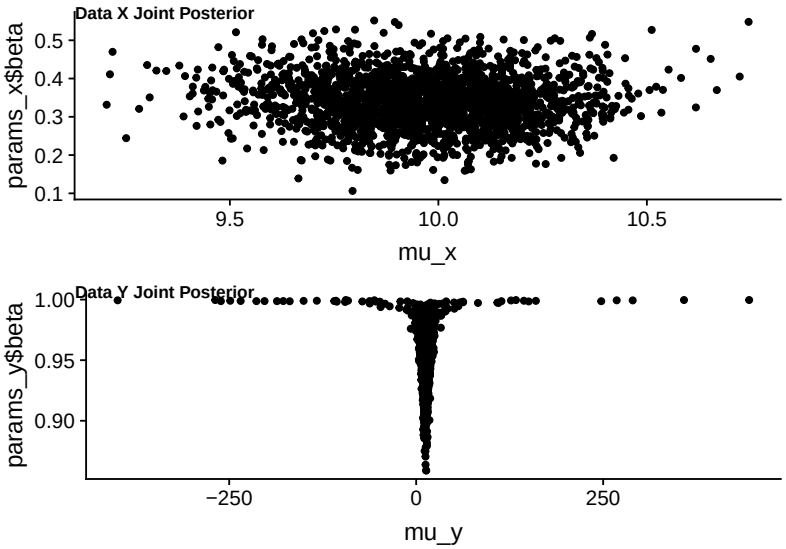
**Traceplots for the fit models**






We can see from the trace plots that the parameters for both the data models have reached convergence.

**Joint Posterior for mu and phi**




From the plot for the joint posterior of mu and phi we can see that the model fit on data X has reached convergence as the values are randomly distributed in the samples. But in the second model, fit on data Y, it looks like it requires some more burn in period. phi looks to have reached convergence but mu has a lot of deviation in it. Simulations for phi for both the models are centered near the true value for the model, which is good.



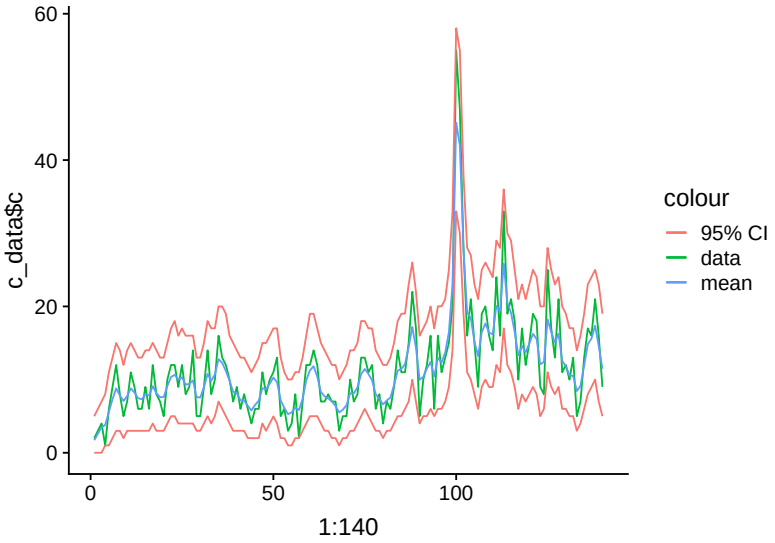
### Question 1C : Stan Model using non-informative prior on campy data

```
stan_code2 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}
"
model <- stan_model(model_code = stan_code2)

c_data <- as.data.frame(read.table("data.txt", header = T))
#hist(c_data$c)
T = length(c_data$c)
```

We are reading the campy data, and creating the stan model for the problem. We are then compiling the model and storing into the variable model. The priors used for this model are non-informative as it is almost flat in the space of the parameter. All the possible values for the parameter are equally likely as we have no prior knowledge about the data.

We are then sampling from this compiled model. We are taking 2 markov chains of 10000 simulations from this posterior with a burnin period of 1000.

**Summary from the model fit on campy data**




Here we calculate the confidence interval for the parameter theta.

theta[t] = exp(x[t])

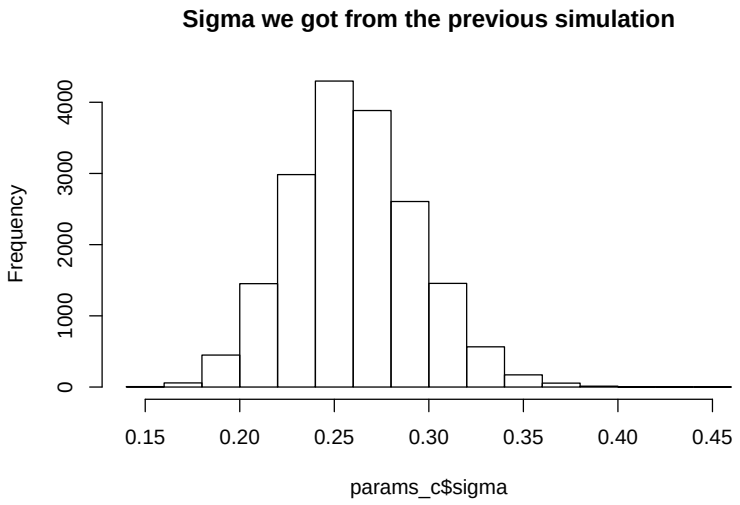
For the calculation of the confidence interval, we simulate from the possion distribution using the theta values. We take 1000 randomly generated values from the poission distribution and calculate the 95% credible interval on it. We do this for all the 140 theta values to get the 95% interval.

For the posterior mean, we take the mean of the randomly generated 1000 samples from the poission distribution and take the mean of that.



### Question 1D : Stan Model using Informative prior on X and Y

```
stan_code3 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(100);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}

"
model <- stan_model(model_code = stan_code3)

hist(params_c$sigma, main="Sigma we got from the previous simulation")
```

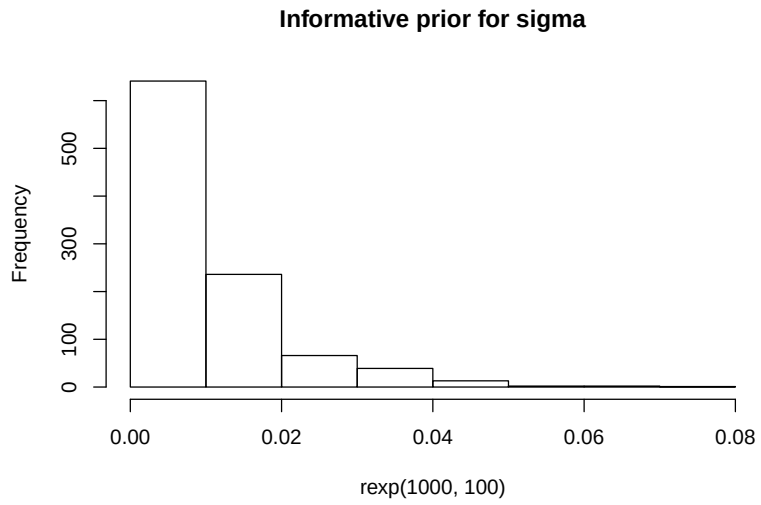



```
mean(params_c$sigma)
```

```
## [1] 0.2606058
```

```
hist(rexp(1000, 100), main="Informative prior for sigma")
```

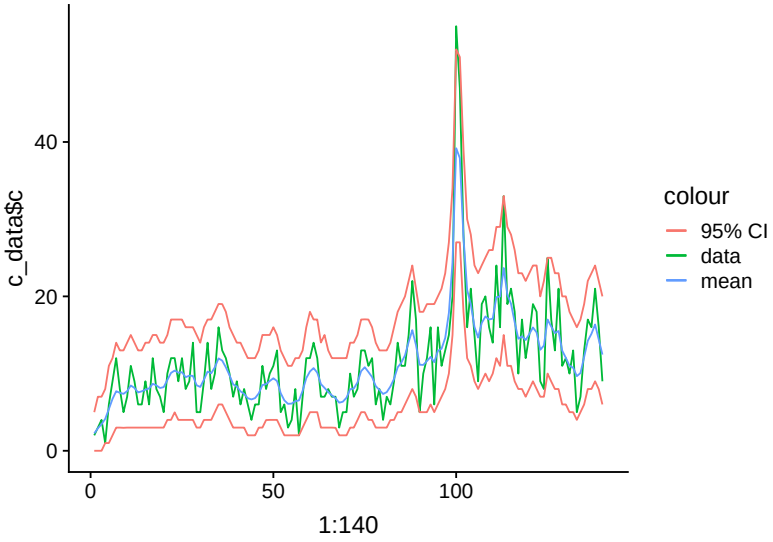




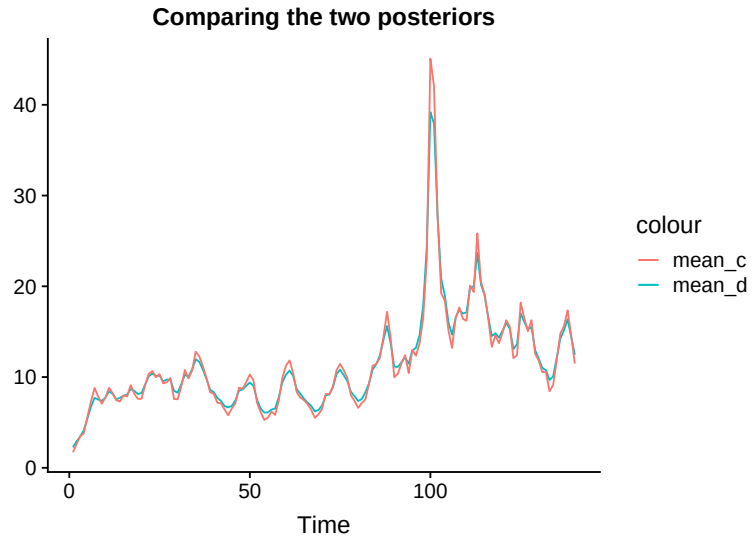

We are creating the stan model for the problem. We are then compiling the model and storing into the variable model. The priors used for this model are informative as we are told that the variance sigma for the data is smaller than the data suggests. The value for sigma we got from the previous model was around 0.25, and we are told that it is smoother than the data suggests. So our prior knowledge tells us that the values of sigma smaller than 0.2 should be given higher probability. We use a exponential distribution to specify the informative prior for sigma. We use a large rate for the exponential distribution so the density is centered close to zero.

We are then sampling from this compiled model. We are taking 4 markov chains of 10000 simulations from this posterior with a burnin period of 1000.

**Summary from fitted model**






The posterior means and CI are calculated in the same manner as we did in the previous question. The posterior mean plot we get using an informative prior is similar, but it is smoother from the one we got in the previous question.

To make a comparision of both, we made a plot of the posterior mean we got from the previous question and the posterior mean calculated using an informative prior in this question. The posterior mean for the informative prior for sigma is a little smoother. This is the only difference we found using a informative prior.

## Appendix

```
library(ggplot2)
library(ggplot2)
library(rstan)
library(cowplot)
options(mc.cores=4)

#Given intial values
mu <- 10
sg_sq <- 2
t_p <- 200   #time points. Number of variables


#function for creating Ar process according to given equation
ar_process <- function(phi,t_p,mu,sg_sq) {
  x <- c()
  x[1] <- mu
  for(i in 2:t_p) {
    x[i] <- mu + phi*(x[i-1]-mu) + rnorm(1,0,sg_sq)
  }

  return(x)
}

#phi value between -1 and 1
phi <- c(-1,0.7,1)

#Creating data for three phi values
ar_data <- do.call(cbind,lapply(phi,ar_process,t_p=200,mu=10,sg_sq=2))
colnames(ar_data) <- c("samp1","samp2","samp3")
ar_data <- as.data.frame(ar_data)

#plotting to check the effect of phi on Ar process. We need not show these three plots.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=samp1,color="samp1"),size=0.9) +
  geom_line(aes(y=samp2,color="samp2"),size=0.9) +
  geom_line(aes(y=samp3,color="samp3"),size=0.9) +
  theme_grey() + scale_color_manual(values = c("#8da0cb", "#fc8d62", "#1b9e77"),
                                    labels = c(phi[1], phi[2], phi[3]))


#choosing 0.7 as  avalue to work further on.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=samp2,color="samp2"),size=0.9)+ theme_grey() +
  scale_color_manual(values = "#fc8d62",labels = phi[2]) +
  ggtitle(expression(paste("AR process with ",phi,"=0.7"))) +
  xlab("Samples") + ylab("AR process")

#1B
stan_code1 = "data {
  int<lower=0> T;
  vector[T] y;
```



```
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  y[2:T] ~ normal(alpha + beta*y[1:(T-1)], sigma);
}
"
#compile model
model <- stan_model(model_code = stan_code1)

#phi value between -1 and 1
phi <- c(0.3,0.95)
T = 200
mu = 10

#Creating data for three phi values
ar_data <- do.call(cbind,lapply(phi,ar_process,t_p=T,mu=mu,sg_sq=2))
colnames(ar_data) <- c("X","Y")
ar_data <- as.data.frame(ar_data)
#head(ar_data)

#plotting to check the effect of phi on Ar process. We need not show these three plots.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=X,color="samp1"),size=0.9) +
  geom_line(aes(y=Y,color="samp2"),size=0.9) +
  scale_color_manual(values = c("#8da0cb", "#fc8d62"),
                     labels = c(phi[1], phi[2]))

#model run on X
fit_x <- sampling(model, list(T=T, y=ar_data$X), iter=1000, warmup=500, chains=4)

#model run on Y
fit_y <- sampling(model, list(T=T, y=ar_data$Y), iter=1000, warmup=500, chains=4)

params_x <- extract(fit_x)
mu_x = params_x$alpha/(1 - params_x$beta)
cat("Mean for mu :", mean(mu_x), "\n\n")
print("95% confidence interval for mu")
quantile(mu_x, probs = c(0.025, 0.975))
cat("\n\n")
plot_grid(ggplot() + geom_density(aes(params_x$alpha)),
          ggplot() + geom_density(aes(params_x$beta)),
          ggplot() + geom_density(aes(params_x$sigma)),
          ggplot() + geom_density(aes(mu_x)))
```

```
#print(fit_x)
a = summary(fit_x)
print("95% confidence intervals for alpha, beta, sigma")
print(a$summary[1:3,c(1, 4, 8, 9)], digits = 4)


params_y <- extract(fit_y)
mu_y = params_y$alpha/(1 - params_y$beta)
cat("Mean for mu :", mean(mu_y), "\n\n")
print("95% confidence interval for mu")
quantile(mu_y, probs = c(0.025, 0.975))
cat("\n\n")
plot_grid(ggplot() + geom_density(aes(params_y$alpha)),
          ggplot() + geom_density(aes(params_y$beta)),
          ggplot() + geom_density(aes(params_y$sigma)),
          ggplot() + geom_density(aes(mu_y)))

#print(fit_y)
b = summary(fit_y)
print("95% confidence intervals for alpha, beta, sigma")
print(b$summary[1:3,c(1, 4, 8, 9)], digits = 4)

plot_grid(nrow = 2, ncol = 1,
          traceplot(fit_x),
          traceplot(fit_y),
          labels = c("Traceplot Model X", "Traceplot Model Y"),label_size = 9)

plot_grid(nrow = 2, ncol = 1,
          ggplot() + geom_point(aes(mu_x, params_x$beta)),
          ggplot() + geom_point(aes(mu_y, params_y$beta)),
          labels = c("Data X Joint Posterior", "Data Y Joint Posterior"),
          label_size = 10)


#1C
stan_code2 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
```



```
  c ~ poisson(exp(x));
}
"
model <- stan_model(model_code = stan_code2)

c_data <- as.data.frame(read.table("data.txt", header = T))
#hist(c_data$c)
T = length(c_data$c)

fit_c <- sampling(model, list(T=T, c=c_data$c), iter=10000, warmup=1000, chains=2)

params_c <- extract(fit_c)

x_post = colMeans(params_c$x)
theta_x = exp(x_post)

post_ci = matrix(0, nrow = 140, ncol = 2)
post_mean = matrix(0, nrow = 140, ncol = 1)
#simulate using this theta_x
for(i in 1:140){
  post_theta = rpois(1000, theta_x[i])
  post_ci[i, ] = quantile(post_theta, probs = c(0.025, 0.975))
  post_mean[i,1] = mean(post_theta)
}

ggplot() +
  geom_line(aes(1:140, c_data$c, col="data")) +
  geom_line(aes(1:140, post_ci[,1], col="95% CI")) +
  geom_line(aes(1:140, post_ci[,2], col="95% CI"))+
  geom_line(aes(1:140, post_mean[,1], col="mean"))


#1D
stan_code3 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(100);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}
```

```
"
model <- stan_model(model_code = stan_code3)

hist(params_c$sigma, main="Sigma we got from the previous simulation")
mean(params_c$sigma)

hist(rexp(1000, 100), main="Informative prior for sigma")

#model run for d
fit_d <- sampling(model, list(T=T, c=c_data$c), iter=10000, warmup=1000, chains=4)

params_d <- extract(fit_d)
x_post_d = colMeans(params_d$x)
theta_x_d = exp(x_post_d)

post_ci_d = matrix(0, nrow = 140, ncol = 2)
post_mean_d = matrix(0, nrow = 140, ncol = 1)
#simulate using this theta_x
for(i in 1:140){
  post_theta = rpois(1000, theta_x_d[i])
  post_ci_d[i, ] = quantile(post_theta, probs = c(0.025, 0.975))
  post_mean_d[i,1] = mean(post_theta)
}

ggplot() +
  geom_line(aes(1:140, c_data$c, col="data")) +
  geom_line(aes(1:140, post_ci_d[,1], col="95% CI")) +
  geom_line(aes(1:140, post_ci_d[,2], col="95% CI"))+
  geom_line(aes(1:140, post_mean_d[,1], col="mean"))


ggplot() +
  geom_line(aes(1:140, post_mean_d[,1], col="mean_d")) +
  geom_line(aes(1:140, post_mean[,1], col="mean_c")) +
  labs(x="Time", y="", title="Comparing the two posteriors")
```