Bayesian Learning: Lab 3

Mohsen Pirmoradiyan, Ahmed Alhasan

2020-05-10

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

The data `rainfall.dat` consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of $\frac{1}{100}$ inch, and records of zero precipitation are excluded) at Snoqualmie Falls, Washington. Analyze the data using the following two models.

(a) *Normal Model*
Assume the daily precipitation $\{y_1, ..., y_n\}$ are independent normally distributed, $y_1, ..., y_n | \mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$ where both μ and σ^2 are unknown. Let $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$ independently of $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$

i. Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\mu, \sigma^2 | y_1, ..., y_n)$. The full conditional posteriors are given on the slides from Lecture 7.

Priors:

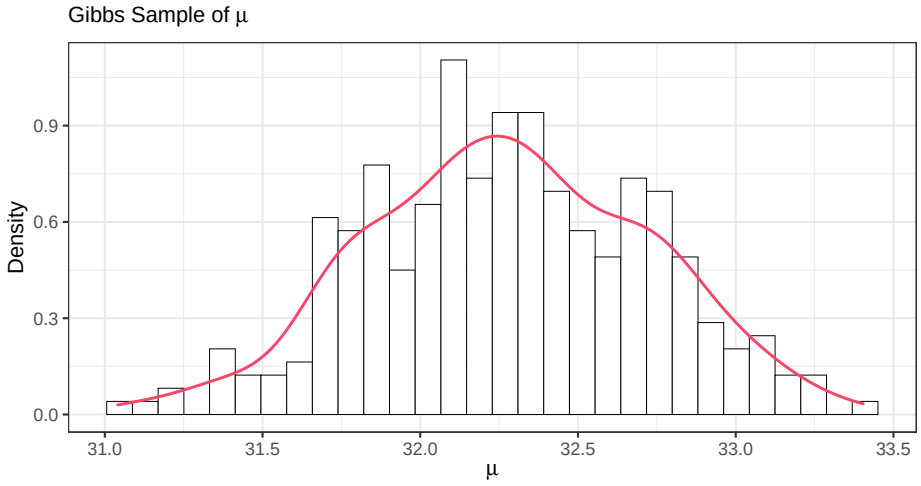
$$\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$$
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

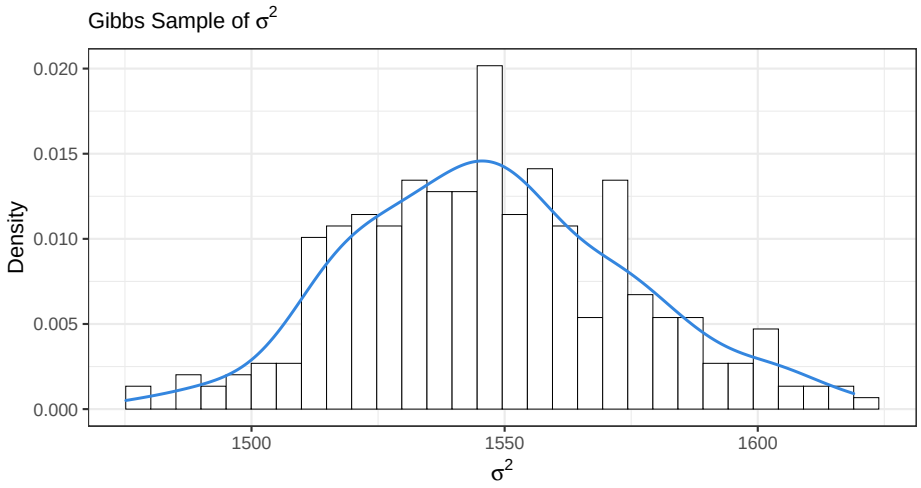
Full Conditional Posteriors:

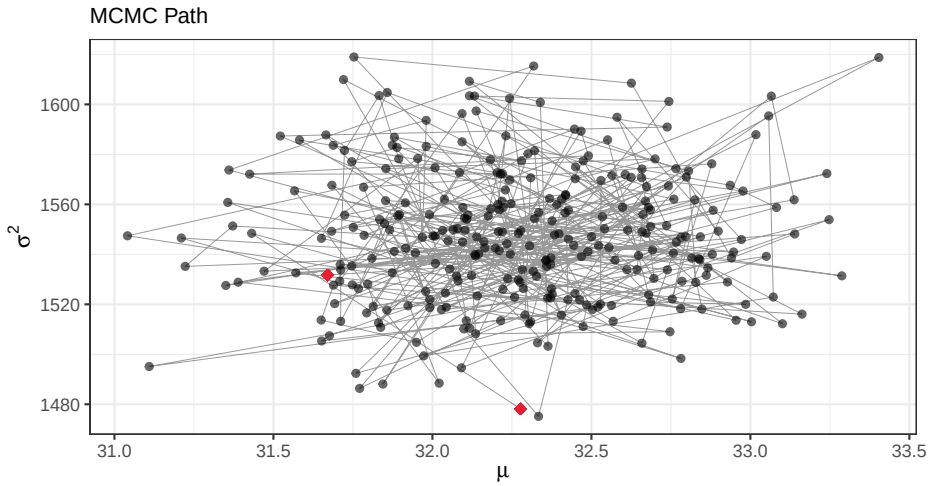
$$\mu | \sigma^2, x \sim \mathcal{N}(\mu_n, \tau_n^2)$$
$$\sigma^2 | \mu, x \sim Inv - \chi^2 \left(v_n, \frac{v_o \sigma_0^2 + \sum_{i=0}^{n} (x_i - \mu)^2}{n + v_0} \right)$$

Where:

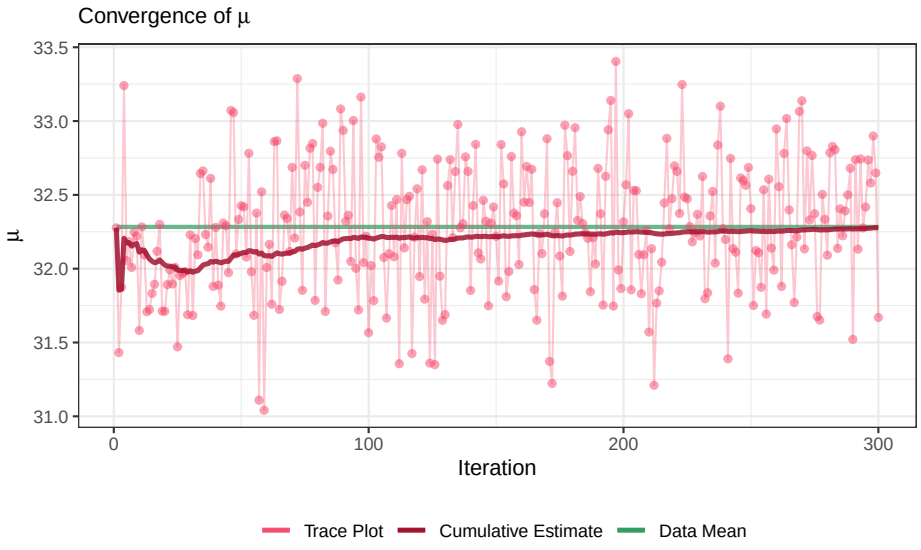
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$
$$w = \frac{n/\sigma^2}{n/\sigma^2 + 1/\tau_0^2}$$
$$\tau_n^2 = \frac{1}{n/\sigma^2 + 1/\tau_0^2}$$
$$\nu_n = \nu_0 + n$$

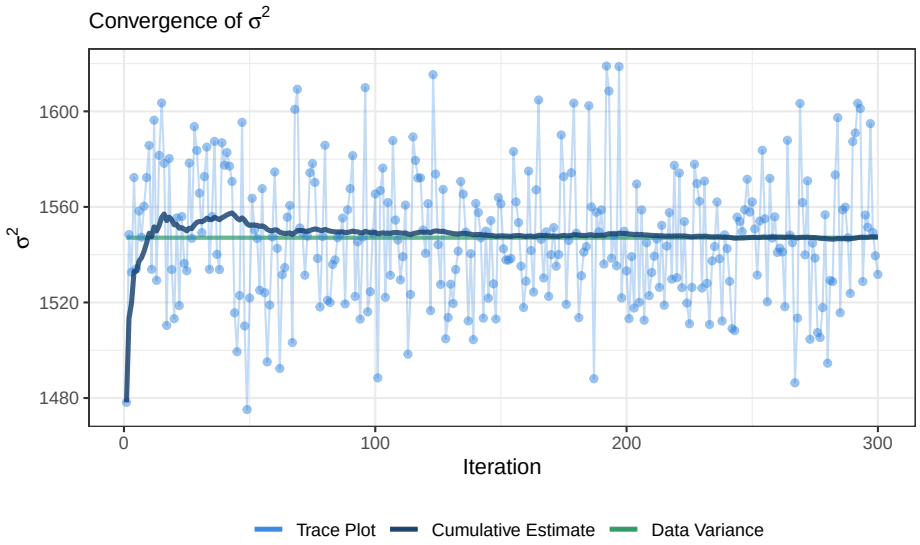






ii. Analyze the daily precipitation using your Gibbs sampler in (a)-i. Evaluate the convergence of the Gibbs sampler by suitable graphical methods, for example by plotting the trajectories of the sampled Markov chains.





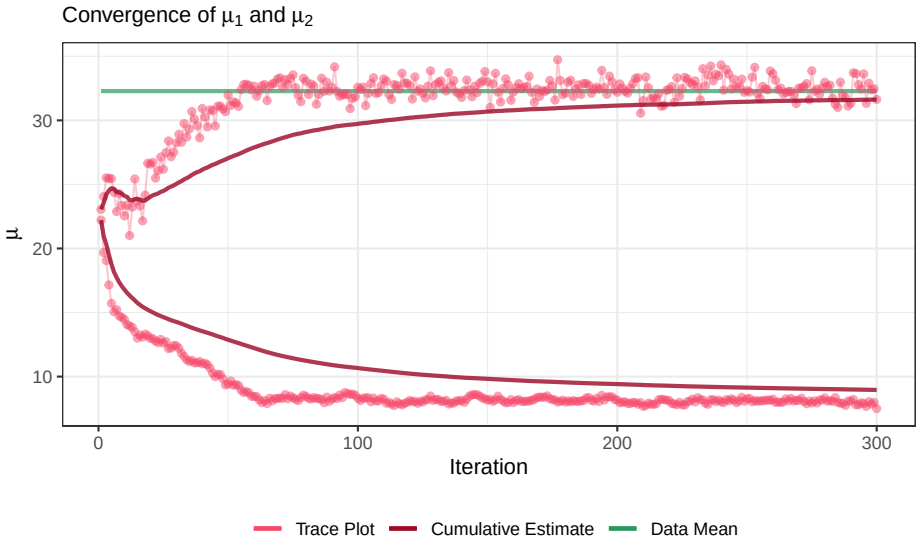
(b) *Mixture normal model.*
Let us now instead assume that the daily precipitation $\{y_1, ..., y_n\}$ follow an iid two-component **mixture of normals** model:

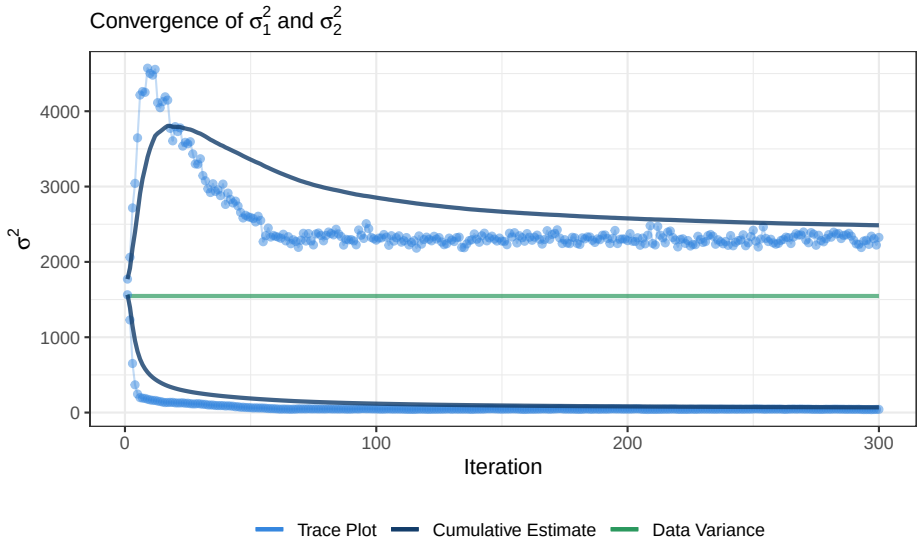
$$p(y_i|\mu, \sigma^2, \pi) = \pi\mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2)$$

where

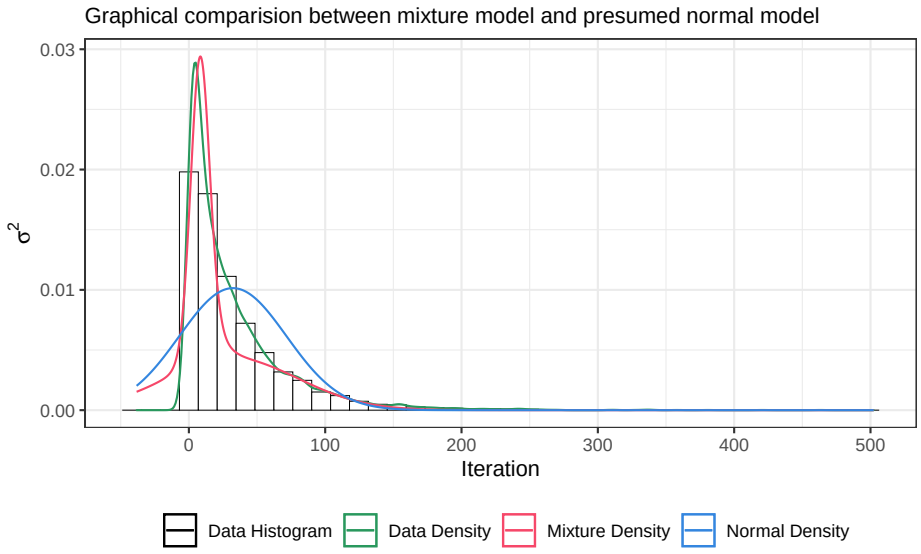
$$\mu = (\mu_1, \mu_2) \text{ and } \sigma^2 = (\sigma_1^2, \sigma_2^2)$$

Use the Gibbs sampling data augmentation algorithm in `NormalMixtureGibbs.R` (available under Lecture 7 on the course page) to analyze the daily precipitation data. Set the prior hyperparameters suitably. Evaluate the convergence of the sampler.





(c) *Graphical comparison.*
Plot the following densities in one figure: 1) a histogram or kernel density estimate of the data. 2) Normal density $\mathcal{N}(y_i|\mu, \sigma^2)$ in (a); 3) Mixture of normals density $p(y_i|\mu, \sigma^2, \pi)$ in (b). Base your plots on the mean over all posterior draws.



2. *Metropolis Random Walk for Poisson regression.*

Consider the following Poisson regression model

$$y_i|\beta \sim \text{Poisson}\left[\exp(x_i^T\beta)\right],\ i = 1, ..., n$$

where y_i is the count for the ith observation in the sample and x_i is the p-dimensional vector with covariate observations for the ith observation. Use the data set `eBayNumberOfBidderData.dat`. This dataset contains observations from 1000 eBay auctions of coins. The response variable is **nBids** and records the number of bids in each auction. The remaining variables are features/covariates (x):

- **Const** (for the intercept)
- **PowerSeller** (is the seller selling large volumes on eBay?)
- **VerifyID** (is the seller verified by eBay?)
- **Sealed** (was the coin sold sealed in never opened envelope?)
- **MinBlem** (did the coin have a minor defect?)
- **MajBlem** (a major defect?)
- **LargNeg** (did the seller get a lot of negative feedback from customers?)
- **LogBook** (logarithm of the coins book value according to expert sellers. Stan- dardized)
- **MinBidShare** (a variable that measures ratio of the minimum selling price (starting price) to the book value. Standardized).

(a) Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: `glm.R`, don't forget that `glm()` adds its own intercept so don't input the covariate Const]. Which covariates are significant?

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

- Significant Covariates with P-value smaller than 0.05

```
##      VerifyID        Sealed       MajBlem       LogBook   MinBidShare
##  1.968202e-05  1.663233e-18  1.571055e-02  3.094651e-05 9.422877e-156
```

(b) Let's now do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim \mathcal{N}[\mathbf{0}, 100 \cdot (X^TX)^{-1}]$ where $\mathbf{X}$ is the $n \times p$ covariate matrix. This is a commonly used prior which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

$$\beta|y \sim \mathcal{N}[\tilde{\beta}, J_y^{-1}(\tilde{\beta})],$$

where $\tilde{\beta}$ is the posterior mode and $J_y(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ can be obtained by numerical optimization (`optim.R`) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

$$P(y|\lambda) = \frac{\lambda^y \cdot e^{-\lambda}}{y!}$$

And given that $y_i|\beta \sim \text{Poisson}\left[\exp(x_i^T\beta)\right]$:

$$P(y|\beta, X) = \frac{e^{y_i(X_i^T\beta)} \cdot e^{-\exp(X_i^T\beta)}}{y_i!} = \frac{e^{y_i(X_i^T\beta) - \exp(X_i^T\beta)}}{y_i!}$$

Getting the likelihood

$$L(\beta|X, y) = \prod_{i=1}^{N} \frac{e^{y_i(X_i^T\beta) - \exp(X_i^T\beta)}}{y_i!}$$

Taking the log of each side we get:

$$\mathcal{L}(\beta|X, y) = \sum_{i=1}^{n} \Big(y_i(X_i^T\beta) - \exp(X_i^T\beta) \Big) - \sum_{i=1}^{n} log(y_i!)$$

And dropping the last term because β does not depend on it

$$\mathcal{L}(\beta|X, y) \propto \sum_{i=1}^{n} \Big(y_i(X_i^T\beta) - \exp(X_i^T\beta) \Big)$$

Table 1: MLE vs MAP

	(Intercept)	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
MLE	1.0724	-0.0205	-0.3945	0.4438	-0.0522	-0.2209	0.0707	-0.1207	-1.8941
	1.0698	-0.0205	-0.3930	0.4436	-0.0525	-0.2212	0.0707	-0.1202	-1.8920

Table 2: Covariance Matrix

	Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
Const	0.0009	-0.0007	-0.0003	-0.0003	-0.0004	-0.0003	-0.0005	1e-04	0.0011
PowerSeller	-0.0007	0.0014	0.0000	-0.0003	0.0001	-0.0002	0.0003	1e-04	-0.0006
VerifyID	-0.0003	0.0000	0.0085	-0.0008	-0.0001	0.0002	0.0003	-3e-04	-0.0004
Sealed	-0.0003	-0.0003	-0.0008	0.0026	0.0004	0.0005	0.0003	-1e-04	-0.0001
Minblem	-0.0004	0.0001	-0.0001	0.0004	0.0036	0.0003	0.0001	1e-04	-0.0001
MajBlem	-0.0003	-0.0002	0.0002	0.0005	0.0003	0.0084	0.0004	-1e-04	0.0003
LargNeg	-0.0005	0.0003	0.0003	0.0003	0.0001	0.0004	0.0032	-3e-04	-0.0001
LogBook	0.0001	0.0001	-0.0003	-0.0001	0.0001	-0.0001	-0.0003	8e-04	0.0010
MinBidShare	0.0011	-0.0006	-0.0004	-0.0001	-0.0001	0.0003	-0.0001	1e-03	0.0051

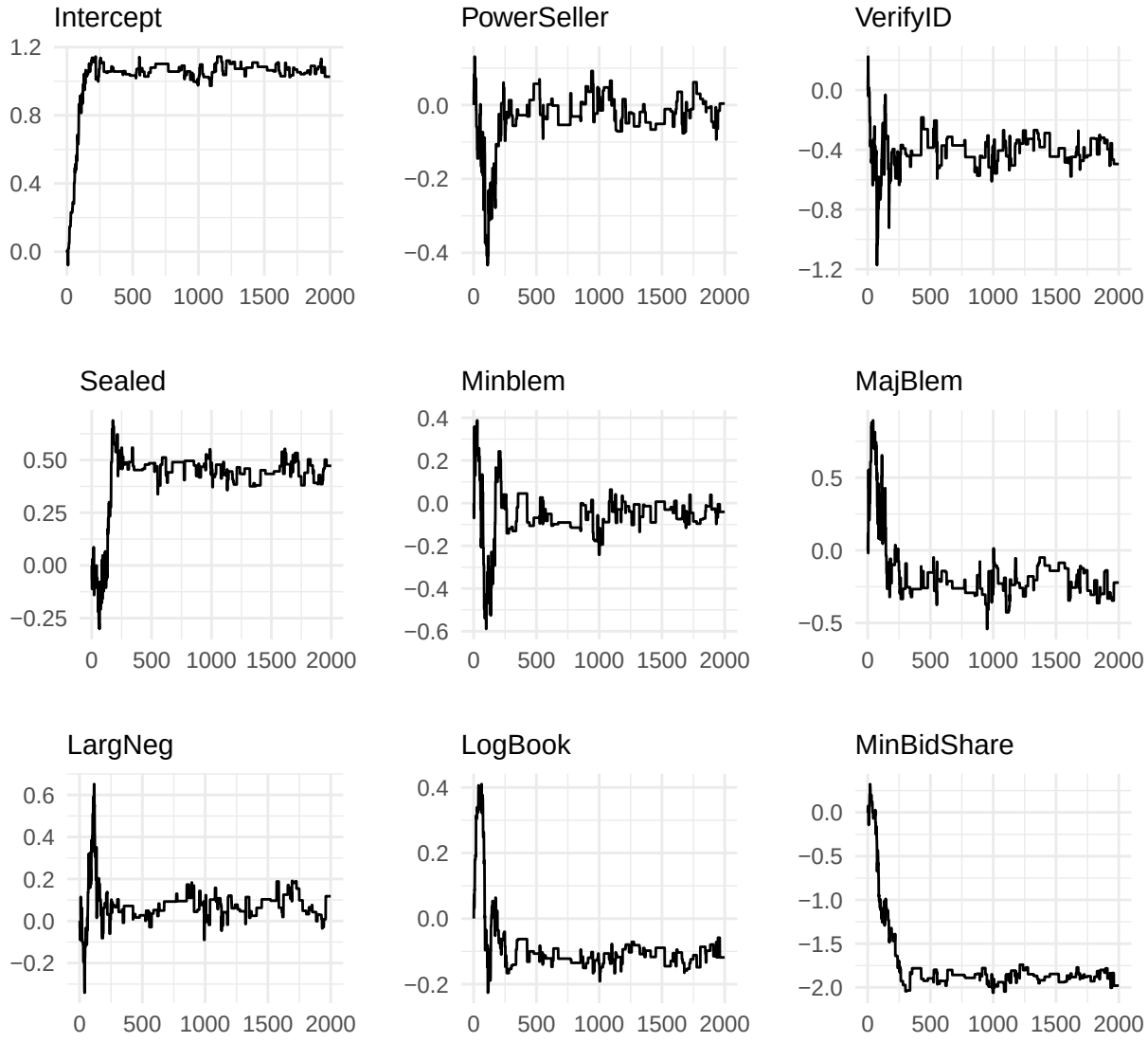
(c) Now, let's simulate from the actual posterior of β using the Metropolis algorithm and compare with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, I will denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

$$\theta_p|\theta^{(i-1)} \sim \mathcal{N}(\theta^{(i-1)}, c \cdot \Sigma),$$

where $\Sigma = J_y^{-1}(\tilde{\beta})$ obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior

density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across *function objects* in R and the triple dot (...) wildcard argument. I have posted a note (`HowToCodeRWM.pdf`) on the course web page that describes how to do this in R.

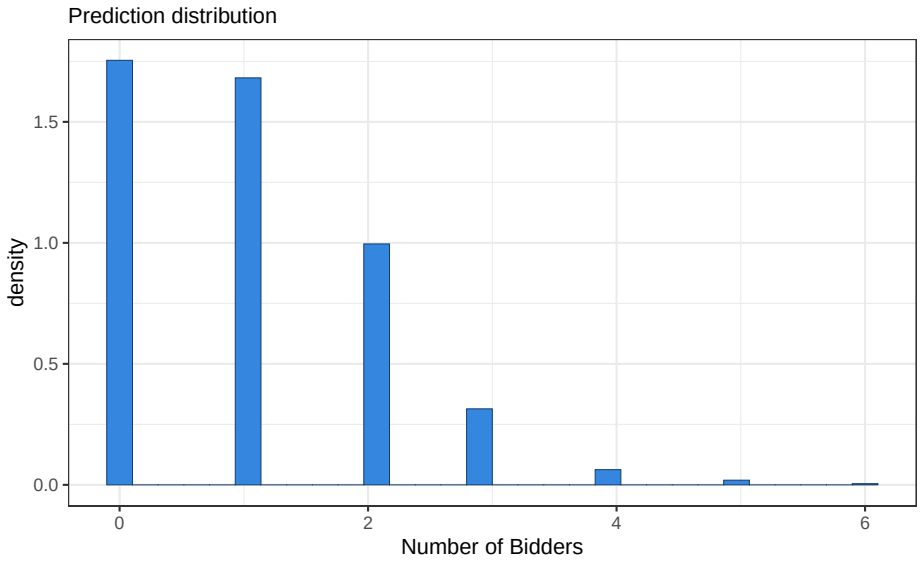
Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.



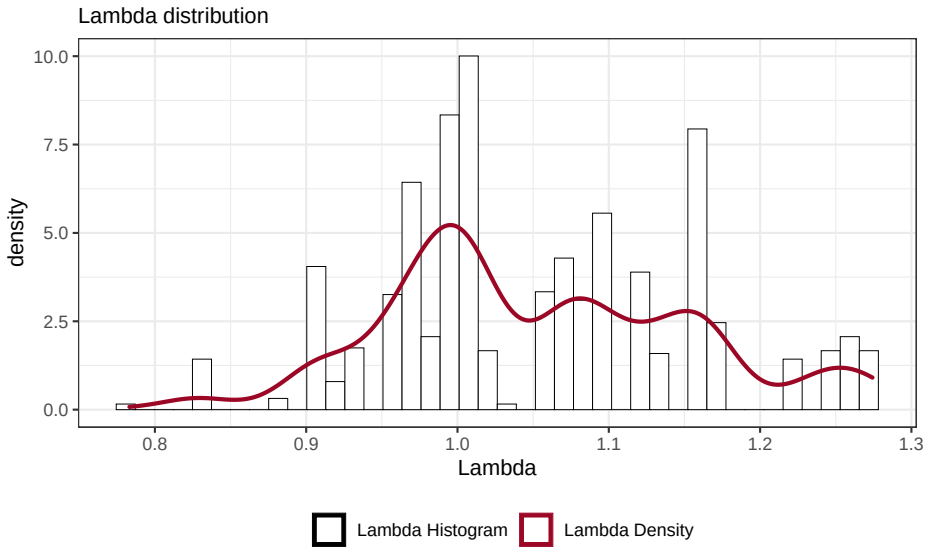
(d) Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?

- **PowerSeller** = 1
- **VerifyID** = 1
- **Sealed** = 1
- **MinBlem** = 0
- **MajBlem** = 0

- **LargNeg** $= 0$
- **LogBook** $= 1$
- **MinBidShare** $= 0.5$




```
## Probability of no bidders in this auction:  0.363
```



Appendix

```
knitr::opts_chunk$set(echo = FALSE, fig.align = "center", warning = FALSE, out.width = "70%", fig.height=4
knitr::read_chunk("R_Code.r")
#setwd("E:/1. Workshop/11. Bayesian Learning/1. Labs/Lab 3")
setwd("C:/Users/WizzCon/Desktop/Machine Learning/1. Workshop/11. Bayesian Learning/1. Labs/Lab 3")
################################################################################
## 1. Normal model, mixture of normal model with semi-conjugate prior.
################################################################################
## 1A_i

rainfall <- read.table("Other/Data/rainfall.dat", header = FALSE)
colnames(rainfall) <- "Precipitation"

n <- dim(rainfall)[1]
mu_0    <- 0
tauSq_0 <- 1
nu_0    <- 1
sdSq_0  <- 1

nDraws <- 300
sdSq_posterior <- 1
Gibbs_sample <- matrix(0,nDraws,2)

for (i in 1:nDraws){
  w <- (n/sdSq_posterior) / ((n/sdSq_posterior) * (1/tauSq_0))
  xbar <- mean(rainfall$Precipitation)
  mu_n <- w * xbar + (1-w) * mu_0
  #mu_n <- xbar + mu_0 ..same as above
  nu_n <- nu_0 + n
  tauSq_n <- (1 / ((n/sdSq_posterior) + (1/tauSq_0)))

  mu_posterior <- rnorm(1, mean = mu_n, sd = sqrt(tauSq_n))
  Gibbs_sample[i,1] <- mu_posterior

  sdSq <- (nu_0 * sdSq_0 + sum((rainfall$Precipitation-mu_posterior)^2)) / nu_n
  sdSq_posterior <- (nu_n * sdSq) / rchisq(1, nu_n)
  Gibbs_sample[i,2] <- sdSq_posterior
}
colnames(Gibbs_sample) <- c("mean", "variance")
Gibbs_sample <- data.frame(x = 1:nDraws, Gibbs_sample)
library(ggplot2)
library("latex2exp")

ggplot(Gibbs_sample) +
  geom_histogram(aes(x = mean, y=..density..), bins = 30, fill = "#ffffffff", colour = "black", size = 0.2
  geom_density(aes(x = mean, y=..density..), color = "#F64769", size = 0.7) +
  labs(subtitle = TeX("Gibbs Sample of $\\mu$"),
       y = "Density",
       x = TeX('$\\mu$')) +
  theme_bw()

ggplot(Gibbs_sample) +
  geom_histogram(aes(x = variance, y=..density..), bins = 30, fill = "#ffffffff",
                 colour = "black", size = 0.2) +
  geom_density(aes(x = variance, y=..density..), color = "#3486DF", size = 0.7) +
  labs(subtitle = TeX("Gibbs Sample of $\\sigma^2$"),
```

```
       y = "Density",
       x = TeX('$\\sigma^2$')) +
  theme_bw()

ggplot(Gibbs_sample) +
  geom_path(aes(x = mean, y = variance), color = "black", alpha = 0.4, size = 0.2) +
  geom_point(aes(x = mean, y = variance), colour = "black", alpha = 0.6) +
  geom_point(aes(x = mean[1], y = variance[1]), colour = "#E91E30", size = 2.5,
             shape = 18) +
  geom_point(aes(x = mean[300], y = variance[300]), colour = "#E91E30", size = 2.5,
             shape = 18) +
  labs(subtitle = "MCMC Path", y = TeX('$\\sigma^2$'), x = TeX('$\\mu$')) +
  theme_bw()
## 1A_ii
cusum_mu <- cumsum(Gibbs_sample[,2])/seq(1,nDraws)
cusum_sdSq <- cumsum(Gibbs_sample[,3])/seq(1,nDraws)
Gibbs_sample <- data.frame(Gibbs_sample, cusum_mu, cusum_sdSq)

ggplot(Gibbs_sample) +
  geom_line(aes(x = x, y = xbar, colour = "Data Mean"), alpha = 0.7, size = 1) +
  geom_point(aes(x = x, y = mean), colour = "#F64769", alpha = 0.5) +
  geom_line(aes(x = x, y = mean, colour = "Trace Plot"), alpha = 0.3) +
  geom_line(aes(x = x, y = cusum_mu, colour = "Cumulative Estimate"), alpha = 0.8, size = 1.2) +
  labs(subtitle = TeX("Convergence of $\\mu$"), x = "Iteration", y = TeX('$\\mu$')) +
  scale_color_manual(breaks = c("Trace Plot", "Cumulative Estimate", "Data Mean"),
                     values = c("#F64769", "#9C0725", "#29985D")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())


ggplot(Gibbs_sample) +
  geom_line(aes(x = x, y = var(rainfall$Precipitation), colour = "Data Variance"), alpha = 0.7, size = 1) +
  geom_point(aes(x = x, y = variance), colour = "#3486DF", alpha = 0.5) +
  geom_line(aes(x = x, y = variance, colour = "Trace Plot"), alpha = 0.3) +
  geom_line(aes(x = x, y = cusum_sdSq, colour = "Cumulative Estimate"), alpha = 0.8, size = 1.2) +
  labs(subtitle = TeX("Convergence of $\\sigma^2$"), x = "Iteration", y = TeX('$\\sigma^2$')) +
  scale_color_manual(breaks = c("Trace Plot", "Cumulative Estimate", "Data Variance"),
                     values = c("#3486DF", "#113B69", "#29985D")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())
## 1B
# Estimating a simple mixture of normals
# Author: Mattias Villani, IDA, Linkoping University. http://mattiasvillani.com

########## BEGIN USER INPUT #################
# Data options
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(1,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 <- 300 # Number of Gibbs sampling draws

# Plotting options
# plotFit    <- TRUE
# lineColors <- c("blue", "green", "magenta", 'yellow')
# sleepTime  <- 0.1 # Adding sleep time between iterations for plotting
################# 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)
  }
  piDraws <- piDraws/sum(piDraws) # Diving every column of piDraws by the sum of the elements in that colu
  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)
}


# Initial value for the MCMC
nObs   <- length(x)
S      <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component a
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 = nObs)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean  <- rep(0,length(xGrid))
effIterCount <- 0
#ylim <- c(0,2*max(hist(x)$density))

mu_records <- matrix(0,nIter,nComp)
colnames(mu_records) <- c("mu1", "mu2")
```

```r
sigma2_records <- matrix(0,nIter,nComp)
colnames(sigma2_records) <- c("sdSq1", "sdSq2")
for (k in 1:nIter){
  #message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the group alloc
  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))
    mu_records[k,j] <- mu[j]
  }

  # 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] + nA
    sigma2_records[k,j] <- sigma2[j]
  }

  # 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 (k%%1 ==0){
    effIterCount <- effIterCount + 1
    #hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k), y
    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
      #lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      #components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    #lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    #legend("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
    #       col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    #Sys.sleep(sleepTime)
  }
```

```
}

# hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted density")
# lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
# lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "blue")
# legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c("


mix_cusum_mu <- apply(mu_records, 2, function(x) cumsum(x)/seq(1,nIter))
colnames(mix_cusum_mu) <- c("mix_cusum_mu1", "mix_cusum_mu2")
mix_cusum_sigma2 <- apply(sigma2_records, 2, function(x) cumsum(x)/seq(1,nIter))
colnames(mix_cusum_sigma2) <- c("mix_cusum_sdSq1", "mix_cusum_sdSq2")
mix_records <- data.frame(x = 1:nIter, mu_records, mix_cusum_mu, sigma2_records, mix_cusum_sigma2)

ggplot(mix_records) +
  geom_line(aes(x = x, y = xbar, colour = "Data Mean"), alpha = 0.7, size = 1) +
  geom_point(aes(x = x, y = mu1), colour = "#F64769", alpha = 0.5) +
  geom_line(aes(x = x, y = mu1, colour = "Trace Plot"), alpha = 0.3) +
  geom_point(aes(x = x, y = mu2), colour = "#F64769", alpha = 0.5) +
  geom_line(aes(x = x, y = mu2, colour = "Trace Plot"), alpha = 0.3) +
  geom_line(aes(x = x, y = mix_cusum_mu1, colour = "Cumulative Estimate"), alpha = 0.8, size = 1) +
  geom_line(aes(x = x, y = mix_cusum_mu2, colour = "Cumulative Estimate"), alpha = 0.8, size = 1) +
  labs(subtitle = TeX("Convergence of $\\mu_1$ and $\\mu_2$"), x = "Iteration", y = TeX('$\\mu$')) +
  scale_color_manual(breaks = c("Trace Plot", "Cumulative Estimate", "Data Mean"),
                     values = c("#F64769", "#9C0725", "#29985D")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())


ggplot(mix_records) +
  geom_line(aes(x = x, y = var(rainfall$Precipitation), colour = "Data Variance"), alpha = 0.7, size = 1) +
  geom_point(aes(x = x, y = sdSq1), colour = "#3486DF", alpha = 0.5) +
  geom_line(aes(x = x, y = sdSq1, colour = "Trace Plot"), alpha = 0.3) +
  geom_point(aes(x = x, y = sdSq2), colour = "#3486DF", alpha = 0.5) +
  geom_line(aes(x = x, y = sdSq2, colour = "Trace Plot"), alpha = 0.3) +
  geom_line(aes(x = x, y = mix_cusum_sdSq1, colour = "Cumulative Estimate"), alpha = 0.8, size = 1) +
  geom_line(aes(x = x, y = mix_cusum_sdSq2, colour = "Cumulative Estimate"), alpha = 0.8, size = 1) +
  labs(subtitle = TeX("Convergence of $\\sigma^2_1$ and $\\sigma^2_2$"), x = "Iteration", y = TeX('$\\sigma
  scale_color_manual(breaks = c("Trace Plot", "Cumulative Estimate", "Data Variance"),
                     values = c("#3486DF", "#113B69", "#29985D")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())
## 1C

ggplot(as.data.frame(x)) +
  geom_histogram(aes(x = x, y=..density.., colour = "Data Histogram"), bins = 40, fill = "#ffffffff", size
  geom_density(aes(x = x, y=..density.., colour = "Data Density")) +
  geom_line(aes(x = xGrid, y = mixDensMean, colour = "Mixture Density")) +
  geom_line(aes(x = xGrid, y = dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), colour = "Normal Density"
  labs(subtitle = "Graphical comparision between mixture model and presumed normal model", x = "Iteration"
  scale_color_manual(breaks = c("Data Histogram", "Data Density", "Mixture Density", "Normal Density"),
                     values = c("black", "#29985D","#F64769", "#3486DF")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())


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

```r
## 2. Metropolis Random Walk for Poisson regression.
################################################################################
## 2A

ebay <- read.table("Other/Data/eBayNumberOfBidderData.dat", header = TRUE)

max_like_est <- glm(nBids ~ ., family = "poisson", data = ebay[,-2])$coefficients
max_like_est
#max_like_est[-1][which(abs(max_like_est[-1]) > 0.3)]

P_values <- summary(glm(nBids ~ ., family = "poisson", data = ebay[,-2]))$coef[,4]
significant <- P_values[-1][which(P_values[-1] < 0.05)]
print(significant)
## 2B

library(mvtnorm)
ebay <- as.matrix(ebay)
X <- ebay[,-1]
y <- ebay[, 1]

mu <- rep(0, ncol(X))
Sigma <- 100 * solve(t(X)%*%X)
initVal <- matrix(rep(0,dim(X)[2]))

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

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

  logLik  <- sum(y*linPred - exp(linPred))
  logPrior <- dmvnorm(as.vector(betaVect), mu, Sigma, log=TRUE)

  return(logLik + logPrior)
}

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

postMode <- OptimResults$par
postCov  <- -solve(OptimResults$hessian)
names(postMode) <- colnames(X)
colnames(postCov) <- rownames(postCov) <- colnames(X)

knitr::kable(t(cbind("MLE" = max_like_est, "MAP" = postMode)), digits = 4, caption = "MLE vs MAP")
knitr::kable(postCov, digits = 4, caption = "Covariance Matrix")
## 2C

RMWSampler <- function(logPostFunc, theta_0, c, Sigma_p, sSize, ...) {

  rejected <- 0
  theta     <- matrix(0, nrow = sSize, ncol = length(theta_0))
  theta[1,] <- theta_0

  for(i in 1:(sSize-1)) {
    proposed <- matrix(rmvnorm(n = 1, mean = theta[i,], sigma = c * Sigma_p))
    alpha    <- min(1, exp(logPostFunc(proposed, ...) - logPostFunc(theta[i,], ...)))
```

```
    u <- runif(1)
    if(u < alpha){
      theta[i+1,] <- proposed
    }
    else{
      theta[i+1,] <- theta[i,]
      rejected    <- rejected + 1
    }
  }
  return(list(sample = theta, rejection_rate = rejected/sSize))
}

theta_0 <- rep(0, dim(X)[2])
sSize   <- 2000
beta_sample <- RMWSampler(logPostFunc = LogPostProbit, theta_0 = theta_0, c = 2, Sigma_p = postCov,
                          sSize = sSize, y = y, X = X, mu = mu, Sigma = Sigma)
colnames(beta_sample$sample) <- colnames(X)
beta_sample_data <- data.frame(Draws = 1:sSize, beta_sample$sample)

p1 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = Const), color = "black") +
  labs(subtitle = "Intercept", x = "", y = "") +
  theme_minimal()

p2 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = PowerSeller), color = "black") +
  labs(subtitle = "PowerSeller", x = "", y = "") +
  theme_minimal()

p3 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = VerifyID), color = "black") +
  labs(subtitle = "VerifyID", x = "", y = "") +
  theme_minimal()

p4 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = Sealed), color = "black") +
  labs(subtitle = "Sealed", x = "", y = "") +
  theme_minimal()

p5 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = Minblem), color = "black") +
  labs(subtitle = "Minblem", x = "", y = "") +
  theme_minimal()

p6 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = MajBlem), color = "black") +
  labs(subtitle = "MajBlem", x = "", y = "") +
  theme_minimal()

p7 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = LargNeg), color = "black") +
  labs(subtitle = "LargNeg", x = "", y = "") +
  theme_minimal()

p8 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = LogBook), color = "black") +
  labs(subtitle = "LogBook", x = "", y = "") +
```

```r
  theme_minimal()

p9 <- ggplot(beta_sample_data) +
  geom_line(aes(x = Draws, y = MinBidShare), color = "black") +
  labs(subtitle = "MinBidShare", x = "", y = "") +
  theme_minimal()

library(gridExtra)
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow = 3)
## 2D
X_new <- c(1, 1, 1, 1, 0, 0, 0, 1, 0.5)

lambdas    <- vector(length = sSize/2)
predSample <- vector(length = sSize/2)
for(pred in 1:(sSize/2)){
  lambdas[pred] <- exp(X_new%*%beta_sample$sample[pred+1000,])
  predSample[pred]  <- rpois(1,lambdas[pred])
}

ggplot(as.data.frame(predSample)) +
  geom_histogram(aes(x = predSample, y=..density.., colour = "Prediction Histogram"),
                 bins = 30, color = "#113B69", fill = "#3486DF", size = 0.2) +
  labs(subtitle = "Prediction distribution",
       x = "Number of Bidders", y = "density") +
  scale_color_manual(breaks = c("Prediction Histogram"),
                     values = c("black")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())


cat("Probability of no bidders in this auction: ", sum(predSample == 0)/ (sSize/2))


getMode <- function(vect){
  # sorted <- sort(vect)
  # tbl    <- table(sorted)
  # n      <- which.max(tbl)
  # predMode_x <- as.numeric(names(tbl)[n])

  dens <- density(vect)
  #ind  <- which(diff(sign(diff(dens$y))) < 0) + 1
  #data.frame(x = dens$x[ind], y = dens$y[ind])
  predMode_x <- dens$x[which.max(dens$y)]
  predMode_y <- max(dens$y)
  return(c(predMode_x,predMode_y))
}
lambdaMode <- getMode(lambdas)

ggplot(as.data.frame(lambdas)) +
  geom_histogram(aes(x = lambdas, y=..density.., colour = "Lambda Histogram"),
                 bins = 40, fill = "#ffffffff", size = 0.2) +
  geom_density(aes(x = lambdas, y=..density.., colour = "Lambda Density"), size = 1) +
  #geom_point(x = lambdaMode[1], y = lambdaMode[2], size = 2) +
  #geom_label(aes(x = lambdaMode[[1]]+0.3, y = lambdaMode[[2]]+0.1), label = "Lambda Mode") +
  labs(subtitle = "Lambda distribution",
       x = "Lambda", y = "density") +
  scale_color_manual(breaks = c("Lambda Histogram", "Lambda Density"),
```

```
                      values = c("black", "#9C0725")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())


# modes <- vector(length = ncol(tbl))
# for(i in 1:ncol(X)){
#   x    <- sort(beta_sample$sample[,i])
#   tbl <- table(x)
#   n    <- which.max(tbl)
#   modes[i] <- as.numeric(names(tbl)[n])
# }
# modes
#
# yPred <- X_new%*%modes
```