PHP 2530: BAYESIAN STATISTICAL METHODS
HOMEWORK I R APPENDIX

NICK LEWIS

Packages

```
#Allows us to use nice functions such as filter
library(dplyr)
#this makes better looking plots in R
library(ggplot2)
library(Cairo) #Windows is bad at makinf good ggplots so this helps with resolution
'
Use this pieces of code to get smooth ggplots for windows. I dont like this
method since it uses too much memory in declaring, but do this to save your plots
'

#ggsave(mixture.plot,path="~/Bayesian Statistical Methods/HW1/Figures",
#       filename = 'Problem 4 PDF Plot.png', dpi = 300, type = 'cairo',
#       width = 11.28, height = 7.38, units = 'in',bg="white")
```

Problem 3

```
### PROBLEM 3 (BDA 3rd. Ed. Exercise 1.9)

poisson.process <- function(theta,time,a,b,num) {
  "
  PARAMETERS
   lambda - rate
   time - time period we're interested in (lambda and time must be same scale)
   a, b - time interval of time spent with patient. i.e. U ~ uniform(a,b)
   num - number of doctors

   Returns:
   -------
    1). Number of arrivals, 2). Number of People who waited, 3). avg wait time,
    4). closing time
  "
  #samples 10*mean(Poisson(lambda*t)) from T ~ Exp(lambda) and sums them.
  #Removes those which exceed the time period
  arr.T <- cumsum(rexp(n=10*time/theta,rate=1/theta)) %>% .[.<= time]
  # records appointment duration wrt opening time
  doc <- rep(0,num)
  wait <- c()
  for(j in 1:length(arr.T) ) {
    # waiting time of patient j
    wait <- c(wait,min(doc) - arr.T[j])
    #appointment duration(if wait>0, appointment starts when doc finishes)
    doc[which.min(doc)] <-ifelse(wait[j]>=0,min(doc),arr.T[j])+runif(1,a,b)
  }
  #if wait <= 0, they didn't wait. If wait > 0, they did
  number.waited <- sum(ifelse(wait > 0, 1, 0))
```

```r
  #waiting time is simply sum of positive waiting times
  time.waiting <- sum(wait[wait > 0])
  #time when office closes
  closing.time <- max(max(doc),time)
  average.wait.time <- ifelse(number.waited==0,0,time.waiting / number.waited)

  ### STORES OUR INFORMATION
  info <- matrix(c(length(arr.T), number.waited,
                   average.wait.time,  closing.time),nrow=1)
  colnames(info) <- c("Number of Arrivals","Number of Patients who Waited",
                      " Average Waiting Time", "Closing Time")
  return(info)
}

poisson.process(theta=10,time=420,a=15,b=20,num=3)


#get 100 samples from the Poisson process.
#use replicate instead of sapply since it keeps the names of the variables
samples <- replicate(100,poisson.process(theta=10,time=420,a=15,b=20,num=3))

apply(samples,2, quantile, probs=c(0.25, .50, .75))
```

Problem 4

```r
### PROBLEM 4 (BDA 3rd. Ed. Exercise 2.4)
y <- seq(0, 300, length=1000)
fy <- function(x, theta){
  dnorm(x, mean = 1000*theta, sd = sqrt(1000*theta*(1-theta)))
}

#calculates fy for each theta giving 1000x3 matrix.
#Then matrix multiplication to give 1000 length vector i.e. (1000x3)(3x1)
p <- outer(y,c(1/12,1/6,1/4),fy) %*% c(0.25,0.5,0.25)

data <- data.frame(y, p)
ggplot(data = data, aes(y, p)) +
  geom_line(color="black",size=1.4)+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab(~ paste(f[Y](y)))+
  xlab("y")

#METHOD 1:

#Note, use 0.9999 for the quantiles in between. 1 gives you Infinity
q <- c(0.20,0.99997,0.50,0.9996,0.80)
mu.theta <- c(1/12,1/12,1/6,1/6,1/4)
```

```r
sd.theta <- sqrt(1000*mu.theta*(1-mu.theta)); mu.theta <- 1000*mu.theta

Q <- sapply(1:length(q),function(x) qnorm(q[x],mu.theta[x],sd.theta[x]))
Q <- matrix(round(Q,3),nrow=1)
colnames(Q) <- c("5%","25%","50%","75%","95%")
print(Q)

#METHOD 2:

# GMQ- Gaussian Mixture Quantiles
GMQ <- function(p,theta,w,y){
  '
    Parameters
    ----------
    p : quantiles we wish to obtian values for
    theta : finite parameter space for theta
    w : weights attached to each theta
    y : upper bound of range to search over (i.e. we look from [0,y])

    Returns
    -------
    Quantiles of the gaussian mixture model

  '

  #functions to use
  gmm <- function(x, theta) {
    pnorm(x, mean = 1000*theta, sd = sqrt(1000*theta*(1-theta)))
    }
  N <- length(p)
  #initialize range to search over
  X <- seq(from=0,to=y,by=0.01);
  G <- (outer(X,theta,gmm) %*% w)
  #finding position of minimum value, then finding
  quantiles <- sapply(1:N,function(x) X[which.min(abs(G - p[x]))])
  quantile.names <-  sapply(1:N,function(x) paste0(100*p[x],"%"))
  #Nice, readable form
  quantiles <- matrix(quantiles,nrow=1); colnames(quantiles) <- quantile.names
  return( quantiles)
}
#quantile values
GMQ(p=c(0.05,0.25,0.50,0.75,0.95),theta=c(1/12,1/6,1/4),w=c(1/4,1/2,1/4),y=500)
```

Problem 6

```r
### PROBLEM 6 (BDA 3rd. Ed. Exercise 2.8)
var.n <- function(n,a,b){ 1/(1/(a)^2 + n/(b)^2)  }
mu.n <- function(n,a,b,m,y){ var.n(n,a,b)*(m/(a)^2 + (n*y)/(b)^2)}

mu <- mu.n(c(10,10,100,100),40,20,180,150)
var <- var.n(c(10,10,100,100),40,20) + c(0,20^2,0,20^2)

sapply(c(0.025,0.975),function(x) qnorm(x,mean=mu , sd =sqrt(var ) ) )
```

Problem 7

```r
### PROBLEM 7 (BDA 3rd. Ed. Exercise 2.10)

#values to sum over
values <- c(203:10000)
#p(X)
p.X <- sum((1/100)*(1/values)*(99/100)^(values - 1))
sprintf("The normalizing constant for the posterior is %.7s",p.X)

#p(N|X)
post <- (1/(100*p.X*values))*(99/100)^(values - 1)
#E(N|X)
mu.N <-  sum(values*post)
sprintf("The posterior mean is %.6s",mu.N)

#Var(N|X) = sum (N-E(N|X))^2 p(N|X)
sd.N <- sqrt(sum((values-mu.N)^2*post))
sprintf("The posterior standard deviation is %.5s",sd.N)

# Part c (Poisson Prior)

#q(N|X), unnormalized posterior
#put everything in terms of log and exponents so R can handle computation
unnorm.post <- exp((values)*log(100)-lfactorial(values)-100-log(values))
#p(X)
p.X1 <- sum(unnorm.post)
#p(N|X)
new.post <- unnorm.post/p.X1
#E(N|X)
mu.N1 <-  sum(values*new.post)
sprintf("The posterior mean is %.6s",mu.N1)

#sd(N|X)
sd.N1 <- sqrt(sum((values-mu.N1)^2*new.post))
sprintf("The posterior standard deviation is %.5s",sd.N1)
```

Problem 8

```r
### PROBLEM 8 (BDA 3rd Ed. Exercise  2.13)
#data for the problem
df <- data.frame(year =1:10 ,
                 accidents=c(24, 25, 31, 31, 22, 21, 26, 20, 16, 22),
                 deaths=c(734, 516, 754, 877, 814, 362, 764, 809, 223, 1066),
                 rate =c(0.19, 0.12, 0.15, 0.16, 0.14, 0.06, 0.13, 0.13, 0.03, 0.15)
)
df["miles"] <- df["deaths"]*(1e8)/df["rate"]

#Prior distribution parameters (here so you can adjust for different priors)
prior.shape <- 0; prior.rate <- 0

## APPROACH 1: FIND POSTERIOR PREDICTIVE DISTRIBUTION

sizes <- c(sum(df["accidents"]),sum(df["accidents"]),
```

```r
        sum(df["deaths"]),sum(df["deaths"])) + prior.shape

#corresponding probability parameters
probs <- c(nrow(df)/(nrow(df)+1+prior.rate),
               sum(df["miles"])/(sum(df["miles"])+(8e11)+prior.rate),
               nrow(df)/(nrow(df)+1+prior.rate),
               sum(df["miles"])/(sum(df["miles"])+(8e11)+prior.rate))

sapply(c(0.025,0.975),function(x) qnbinom(x,size = sizes,prob= probs ) )

## APPROACH 2: SAMPLE FROM POSTERIOR, PLUG BACK INTO LIKELIHOOD
#Strategy:
#(sample from theta|y, plug values into y|theta, sort from least to greatest)
#find 25th and 975th values, these represent endpoints of 95% posterior interval

#shape, rate and miles
a <- c(sum(df["accidents"]),sum(df["accidents"]),
      sum(df["deaths"]),sum(df["deaths"])) + prior.shape

b <- c(nrow(df),sum(df["miles"]),nrow(df),sum(df["miles"])) + prior.rate

m <- c(1,8e11,1,8e11)

#shapes
sapply(1:4,function(x) sort(rpois(1000,m[x]*rgamma(1000,a[x],b[x])))[c(25,975)])
```