TDDE07_20210603

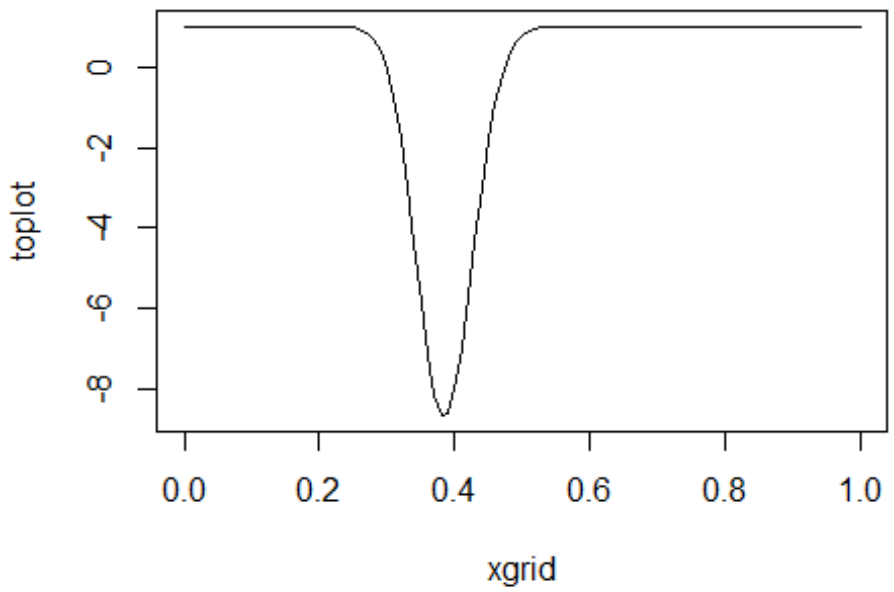
Exam TDDE07 20210603 Daniel Bissessar 9811203893 danbi675

Question 1

```
#a)
n <- 100
sa <- 38
alpha <- 16
beta <- 24
prob <- pbeta(0.4,alpha+sa, beta+n-sa)
print(c("Posterior probability is ", prob))
```

```
## [1] "Posterior probability is " "0.639961700759514"
```

```
xgrid <- seq(0,1,0.01)
toplot <- 1-dbeta(xgrid,alpha+sa, beta+n-sa)
plot(xgrid,toplot, type = 'l')
```




```
#b)
draws <- rbeta(10000, alpha+sa, beta+n-sa)
draws <- (1-draws)/draws
```

```
interval <- quantile(draws, probs = c(0.025,0.975))
print(interval)
```

```
##     2.5%    97.5% 
## 1.139428 2.260659
```

```
#The odds of a customer buying product a from the selection are in the interv
al 1.14:1 to 2,25:1

#c) From exercise session 4
mlikelihood <- beta(alpha+sa, beta+n-sa)/beta(alpha,beta)
print(c("Marginal likelihood: ",mlikelihood))
```

```
## [1] "Marginal likelihood: " "7.55677069331938e-30"
```

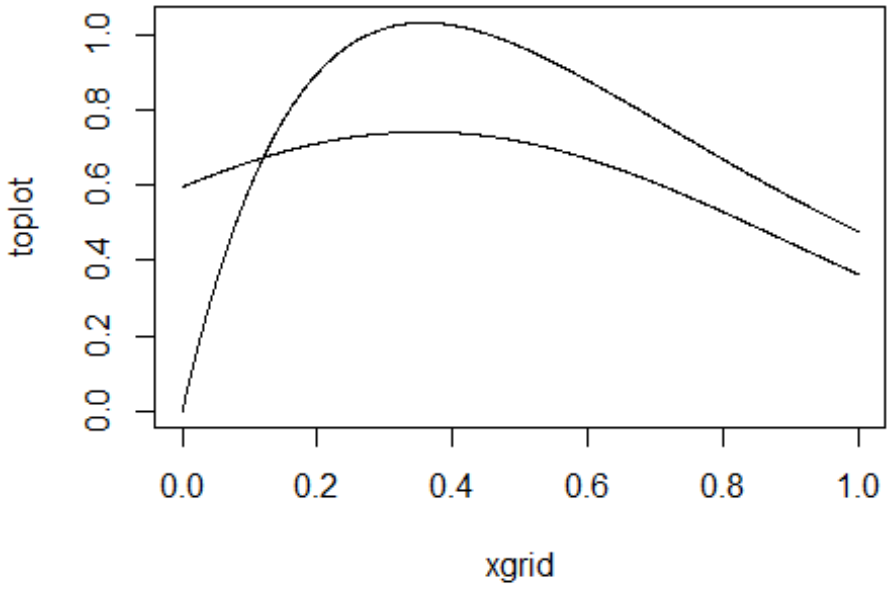
```
#d)
dirichletpar <- c(20+38,20+27,20+35)
library(DirichletReg)

draws <- rdirichlet(10000, dirichletpar)
prob <- sum(draws[,1]>draws[,3])/10000
print(c("The posterior probability is :", prob))
```

```
## [1] "The posterior probability is :" "0.6056"
```

Question 2

For a,b,c see PaperSol.pdf

```
#d)
logpostdist <- function(x,samp) {
  sampsumsqr <- sum(samp**2)
  return(dgamma(x,2,rate = sampsumsqr))
}
xgrid <- seq(0,1,0.001)
toplot <- rep(0, length(xgrid))
count <- 1
for (i in xgrid) {
  toplot[count] <- logpostdist(i,sqrt(2.8))
  count <- count + 1
}
plot(xgrid,toplot, type = 'l')
#e)
fxtheta <- function(theta, x) {
  2*theta*x*exp(-theta*x^2)
}
approx <- optim(0.5, fxtheta, x = sqrt(2.8), lower = 0.1, method = 'L-BFGS-B'
, hessian = TRUE, control = list(fnscale=-1))
count <- 1
approxplot <- rep(0, length(xgrid))
for (i in xgrid) {
  approxplot[count] <- dnorm(i,approx$par,sqrt(-(solve(approx$hessian))))
  count <- count + 1
}
lines(xgrid, approxplot)
```


toplot
0.0
0.2
0.4
0.6
0.8
1.0
xgrid
0.0
0.2
0.4
0.6
0.8
1.0

Question 3

```
# Reading the data from file
load(file = 'C:/Users/Daniel Bissessar/Downloads/UniversityEntrance.RData')
library(geoR)

library(mvtnorm)
BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones if y
ou want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCovs m
atrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 vecto
r

  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%mu_0
- t(mu_n)%*%Omega_n%*%mu_n))/v_n)
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rinvchisq(n=1, df=v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2
```

```
    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample))
}
#a)
n <- 10000
mu0 <- c(0,0,0,0,0,0,0)
v0 <- 1
sigma20 <- 4
omega0 <- 25*diag(7)


samps <- BayesLinReg(y,X,mu0,omega0,v0,sigma20,n)
betas <- samps$betaSample
b0 <- c(mean(betas[,1]), quantile(betas[,1], probs = c(0.025,0.975)))
b1 <- c(mean(betas[,2]), quantile(betas[,2], probs = c(0.025,0.975)))
b2 <- c(mean(betas[,3]), quantile(betas[,3], probs = c(0.025,0.975)))
b3 <- c(mean(betas[,4]), quantile(betas[,4], probs = c(0.025,0.975)))
b4 <- c(mean(betas[,5]), quantile(betas[,5], probs = c(0.025,0.975)))
b5 <- c(mean(betas[,6]), quantile(betas[,6], probs = c(0.025,0.975)))
b6 <- c(mean(betas[,7]), quantile(betas[,7], probs = c(0.025,0.975)))
print(c("Beta 0: ", b0))

##                                                          2.5%
97.5%
##          "Beta 0: "  "1.04694044181316" "0.897426691756313"  "1.1951667551
2569"

print(c("Beta 1: ", b1))

##                                                          2.5%
97.5%
##          "Beta 1: " "0.510123713633544" "0.354244222698317" "0.66488495591
2163"

print(c("Beta 2: ", b2))

##                                                              2.5%
##           "Beta 2: "  "0.207783071897359" "0.0843370759000889"
##                97.5%
##  "0.331257597432662"

print(c("Beta 3: ", b3))

##                                                              2.5%
##           "Beta 3: "   "0.22692250763477" "-0.036403996116405"
##                97.5%
##  "0.492287321008506"
```

```
print(c("Beta 4: ", b4))
```

```
##                                                                 2.5%
##           "Beta 4: " "0.0865530366427908" "-0.100619103282906"
##                97.5%
##   "0.27803039420171"
```

```
print(c("Beta 5: ", b5))
```

```
##                                                                 2.5%
##           "Beta 5: "  "0.101021803416884" "-0.160555648015241"
##                97.5%
##  "0.359951612938553"
```

```
print(c("Beta 6: ", b6))
```

```
##                                                                     2.5%
##            "Beta 6: " "-0.0508552213614252"  "-0.239726977339611"
##                 97.5%
##   "0.140132684097251"
```

```
#The strictly positive values for the interval of beta1 can be interpreted as
verbal IQ being a factor that increases the score on a university exam

#b)
postmed <- median(samps$sigma2Sample)
print(postmed)
```

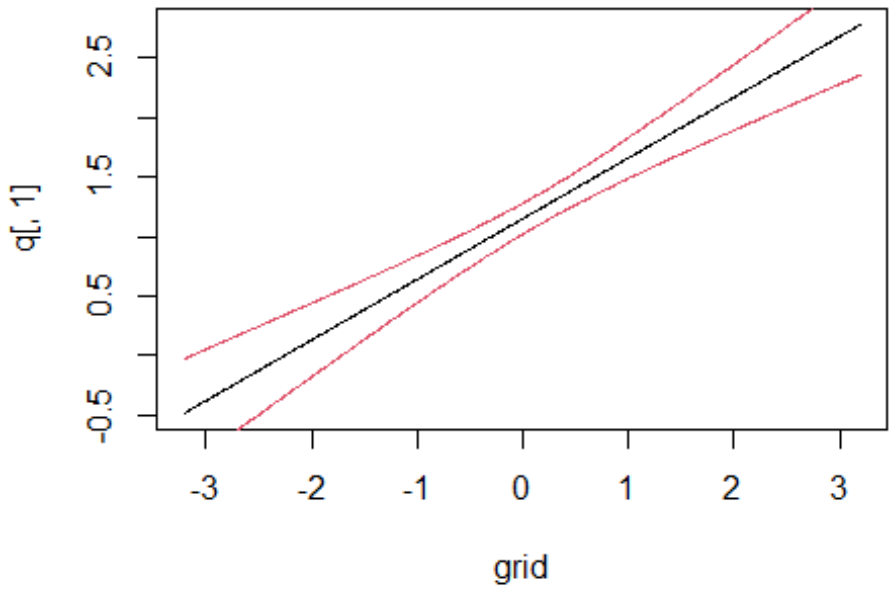
```
## [1] 0.6673574
```

```
#c) to compare these we can compare the distributions for beta5 and beta6 wit
h a Z-test to see if they have the same distribution
beta5mean <- mean(betas[,6])
beta6mean <- mean(betas[,7])
sdbeta5 <- sd(betas[,6])
sdbeta6 <- sd(betas[,7])
z <- (beta5mean-beta6mean)/sqrt(sdbeta5**2+sdbeta6**2)
print(z)
```

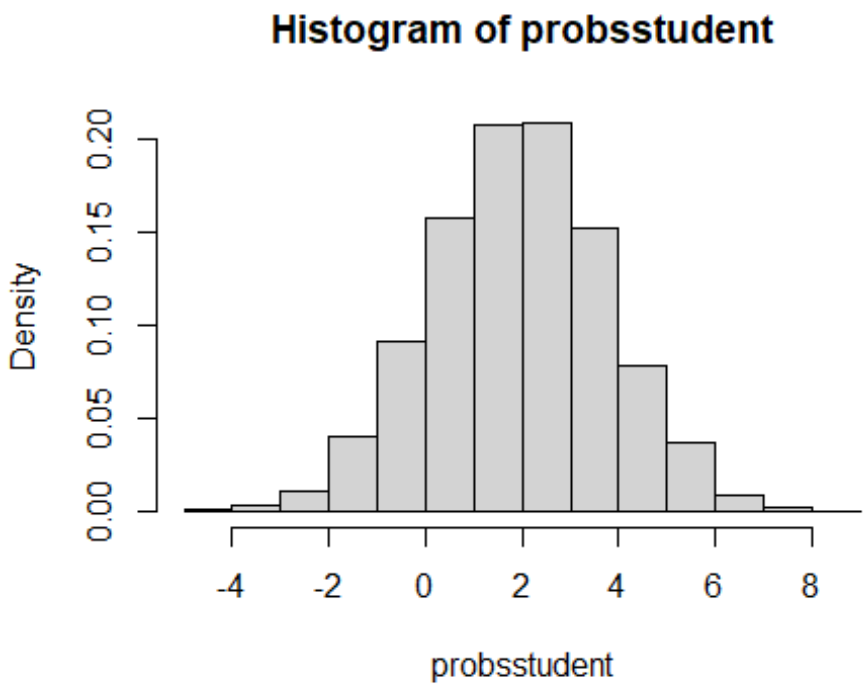
```
## [1] 0.924068
```

```
#since z < 2 we cannot say that beta5 and beta6 are different

#d)
gridstep <- 0.01
start <- min(X[,2])
stop <- max(X[,2])
grid <- seq(start,stop,gridstep)
known <- c(0.5,0,0)
probFunc1 <- function(grid, x){
  y = betas[,1] + betas[,2]*grid + betas[,3]*x[1] + betas[,4]*x[2] + betas[,5
]*x[3] + betas[,6]*grid*x[2] + betas[,7]*grid*x[3]
  quants <- quantile(y, probs = c(0.05,0.95))
```

```
  return(c(mean(y),quants[1],quants[2]))
}
q <- matrix(0, length(grid), 3)
count <- 1
for (i in grid) {
  q[count,] <- probFunc1(i,known)
  count <- count + 1
}
plot(grid,q[,1], type = 'l')
lines(grid,q[,2], col = 2)
lines(grid,q[,3], col = 2)
```




```
#e)
student <- c(0.4,1,1,0)

probFunc2 <- function(x, n){
  be <- rmvnorm(n, c(mean(betas[,1]),mean(betas[2]),mean(betas[,3]),mean(beta
s[,4]),mean(betas[,5]),mean(betas[,6]),mean(betas[,7]), postmed))
  y = be[,1] + be[,2]*x[1] + be[,3]*x[2] + be[,4]*x[3] + be[,5]*x[4] + be[,6]
*x[1]*x[3] + be[,7]*x[1]*x[4]

  return(y)
}
probsstudent <- probFunc2(student, n)
hist(probsstudent, freq = F)
```


Histogram of probsstudent
Density
0.00
0.05
0.10
0.15
0.20
-4
-2
0
2
4
6
8
probsstudent