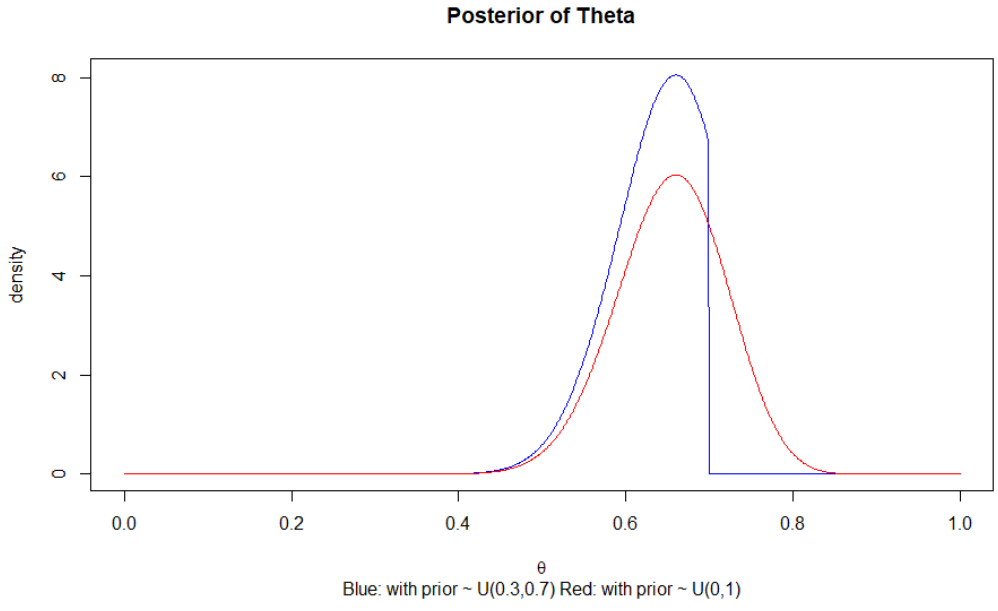
Exam Tdde07 2020 06/04

1 b)

We use the known density functions over a grid of thetas to calculate the posterior distributions for each prior. We normalize by multipliying with 1/gridWidth and dividing by the sum of the posterior. This to ensure that sum(gridWidth*posterior) = 1. +

```
x = 33

gridWidth = 0.001
thetaGrid = seq(0, 1, by=gridWidth)
posterior = rep(0 , length(thetaGrid))
posteriorOld = rep(0 , length(thetaGrid))


for(i in 1:length(thetaGrid)){

  posterior[i] = dbeta(thetaGrid[i],x+1,51-x)*dunif(thetaGrid[i],0.3,0.7)
  posteriorOld[i] = dbeta(thetaGrid[i],x+1,51-x)
}

normPosterior = 1/gridWidth * posterior/sum(posterior)
posteriorOld = 1/gridWidth * posteriorOld/sum(posteriorOld)

plot(x=thetaGrid, y = normPosterior, type="l", xlab=expression(theta), ylab="d
ensity", main="Posterior of Theta", sub="Blue: with prior ~ U(0.3,0.7) Red: wi
th prior ~ U(0,1)", col="blue")
lines(x=thetaGrid, y=posteriorOld, col="red")
```



Uniform prior cuts off the blue posterior at 0.7.

1 c)

We can compute the probability by summarizing the posteriors from 1 to the index of theta=0.5 from thetaGrid. We need to multiply by thetaGrid to get the correct approximation of the area under the curve, as the values in the arrays are discrete approximations of the posterior.

```
indexTheta = which(thetaGrid == 0.5)

p1 = sum(gridWidth*normPosterior[1:indexTheta-1])
p2 = sum(gridWidth*posteriorOld[1:indexTheta-1])
```

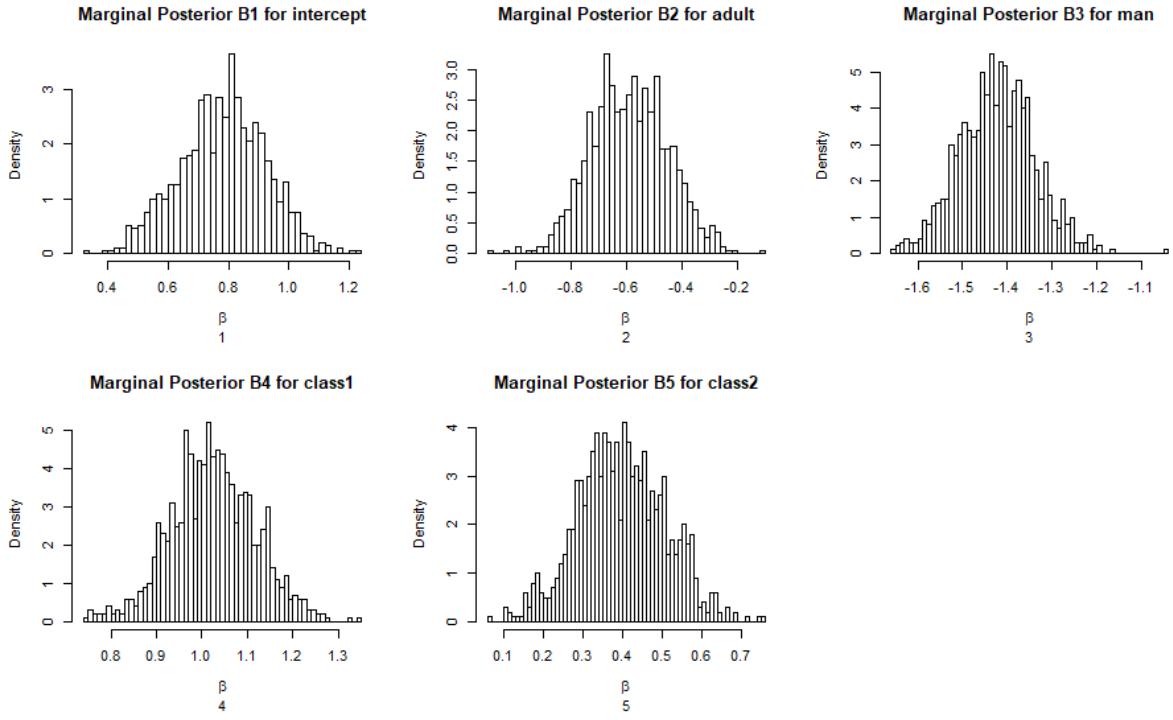
```
> p1
[1] 0.01576212
> p2
[1] 0.01182461
```

pr(theta<0.5) when theta ~ U(0.3,0.7) =~ 1.6%

pr(theta<0.5) when theta ~ U(0.3,0.7) =~ 1.2%

2 a)

```
# a)

X = subset(titanic, select = -survived)
y = titanic$survived

ncov = ncol(X)
mu_0 = rep(0,ncov)
tau = 50
nIter = 1000

draws = BayesProbReg(y, as.matrix(X), mu_0, tau, nIter)

par(mfrow=c(2,3))
hist(draws[,1],50, probability = TRUE,xlab=expression(beta),sub=1, main="Margi
nal Posterior Beta 1")
hist(draws[,2],50, probability = TRUE,xlab=expression(beta),sub=2, main="Margi
nal Posterior Beta 2")
hist(draws[,3],50, probability = TRUE,xlab=expression(beta),sub=3, main="Margi
nal Posterior Beta 3")
hist(draws[,4],50, probability = TRUE,xlab=expression(beta),sub=4, main="Margi
nal Posterior Beta 4")
hist(draws[,5],50, probability = TRUE,xlab=expression(beta),sub=5, main="Margi
nal Posterior Beta 5")
```



2 b)

The point estimator for theta given linear loss is the posterior median (lecture 4).

```
# b)
bMedians = apply(draws,2,median)
```

```
> bMedians
[1]  0.7812789 -0.5884875 -1.4264133  1.0394590  0.3973971
```

2 c)

```
# c)
sum(draws[,2]+ draws[,5]>0)/nrow(draws)
```

```
> sum(draws[,2]+ draws[,5]>0)/nrow(draws)
[1] 0.105
```

These are b2 for adult and b5 for class2. A probability of 10% for these betas simultaneously implies that an adult from second class is a combination that decreases the likelihood for survival, as these two betas sums to a negative response variable with 90% probability.

3 a)

```
# a)

y = c(5,3,17,8)
x = log(c(20,20,50,40))


logPost = function(b,x,y){
  logPrior = dnorm(b, 1, 1/10, log = T)
  logLike = sum(dpois(y,exp(x*b), log = T))
  return(logPrior+logLike)
}

betaInit = 0

optRes<-optim(betaInit, logPost,gr=NULL,x,y,method="L-BFGS-
B",control=list(fnscale=-1),hessian=TRUE)

betaMode = optRes$par
var = -1/optRes$hessian



samples = 10000
samples = rnorm(samples,betaMode, sqrt(Var))
```

```
> betaMode
[1] 0.687575
> sd
             [,1]
[1,] 0.03958641
```

3 b)

We need to find x5 such that $min_{x5}E_{\beta}\big(L(y5,x5)\big)$ $x5 \in (\log(20), \log(40))$ $y5 \sim Poisson(x5 * \beta)$.

We do this by taking 10000 samples from our estimated posterior distribution of beta from a). For each sample of beta, and for each x5, we calculate the loss N times. We take the mean of all losses to find the expected loss for that x5.

```
# b)

loss = function(y,x){
  return(4 + exp(x)/50-sqrt(y))
}

expLoss = function(x, Betas, nPreds){
  losses1 = matrix(0,length(Betas),nPreds)
```

```
  # for each beta, do nPreds from rpois and save in row in losses matrix
  for(i in 1:length(Betas)){
    losses1[i,] = loss(rpois(1,Betas[i]*x),x)
  }
  return(mean(losses1))
}

# minimize expected loss over a E(L(20)) and E(L(40))


# draWS from posterior
Betas = rnorm(10000,betaMode, sd)
# draws from Poisson for each draw of posterior
nPreds = 1000


# compute expected loss for logx = 20

Eloss1 = expLoss(log(20),Betas,nPreds)


# compute expected loss for logx5 = 40

Eloss2 = expLoss(log(40),Betas,nPreds)

print(sprintf("E(Loss(20)) = %g E(Loss(40)) = %g", Eloss1, Eloss2))
```

```
[1] "E(Loss(20)) = 3.11607 E(Loss(40)) = 3.32182"
```

We see from the print that there is a slight difference from the two loss function's expected values. The expected value for spending 20 million has an about 0.2 lower expected loss. Therefor the Bayesian decision would be to cut spendings to 20 million.

Further analysis could be made for risk aversion by studying difference in variance between the model, and/or different probabilities of the loss function, e.g prob(Loss(40)>10) vs prob(Loss(20)>10). This could lead to a more sophisticated decision.

4 c)

```
#P(L|M)
# L: length, meanxML: avarage length of male, MUseq: sequence of mus to integr
ate over
# gridWidth: width of grid in MUseq
pLgivenM = function(L, meanxML, MUseq, gridWidth){


  # to estimate p(L|M) =  Integral( p(L| mu)*p(mu|xmean) dmu )
  # calc products for each mu in MUseqin loop, save in TempInt and  multiply w
ith
  # width of grid. When all values are computed, summarize to get integral.
  tempInt = rep(0, length(MUseq))
  for(i in 1:length(MUseq)){

    #p(L| mu)
    posterior = dnorm(MUseq[i],meanxML,2/sqrt(4))

    #p(mu|xmean)
    likelihood = dnorm(L,MUseq[i],2)

    tempInt[i] = gridWidth*posterior*likelihood

  }

  return(sum(tempInt))
}


#P(W|M)
# W: weight, meanxMW: avarage weight of male, MUseq: sequence of mus to integr
ate over
# gridWidth: width of grid in MUseq
pWgivenM = function(W,meanxMW, MUseq, gridWidth){



  # to estimate p(W|M) =  Integral( p(W| mu)*p(mu|xmean) dmu )
  # calc products for each mu in MUseq in loop, save in TempInt and  multiply
with
  # width of grid. When all values are computed, summarize to get integral.
  tempInt = rep(0, length(MUseq))
  for(i in 1:length(MUseq)){

    #p(W| mu)
```

```
    posterior = dnorm(MUseq[i],meanxMW,50/sqrt(4))

    #p(mu|xmean)
    likelihood = dnorm(W,MUseq[i],50)

    tempInt[i] = gridWidth*posterior*likelihood
  }

  return(sum(tempInt))
}

# P(L|F)
# L: length, meanxFL: avarage length of female, MUseq: sequence of mus to inte
grate over
# gridWidth: width of grid in MUseq
pLgivenF = function(L,meanxFL, MUseq, gridWidth){

  # to estimate p(L|F) =  Integral( p(L| mu)*p(mu|xmean) dmu )
  # calc products for each mu in MUseq  in loop, save in TempInt and  multiply
 with
  # width of grid. When all values are computed, summarize to get integral.
  tempInt = rep(0, length(MUseq))
  for(i in 1:length(MUseq)){

    #p(L| mu)
    posterior = dnorm(MUseq[i],meanxFL,2/sqrt(16))

    #p(mu|xmean)
    likelihood = dnorm(L,MUseq[i],2)

    tempInt[i] = gridWidth*posterior*likelihood
  }

  return(sum(tempInt))
}



# P(W|F)
# W: weight, meanxFW: avarage weight of female, MUseq: sequence of mus to inte
grate over
# gridWidth: width of grid in MUseq
pWgivenF = function(W, meanxFW, MUseq, gridWidth){

  # to estimate p(W|F) =  Integral( p(W| mu)*p(mu|xmean) dmu )
  # calc products for each mu in MUseq  in loop, save in TempInt and  multiply
 with
  # width of grid. When all values are computed, summarize to get integral.
  tempInt = rep(0, length(MUseq))
```

```
  for(i in 1:length(MUseq)){


    #p(W| mu)
    posterior = dnorm(MUseq[i],meanxFW,50/sqrt(16))

    #p(mu|xmean)
    likelihood = dnorm(W,MUseq[i],50)

    tempInt[i] = gridWidth*posterior*likelihood
  }

  return(sum(tempInt))
}

# classifies given length and weight and previous data
naiveBayesClassifier = function(L, W){
  # from b) pF = 3/4 pM = 1/4
  pF = 3/4
  pM = 1/4

  weightGridWidth = 0.01
  weightSeq = seq(150,450,by=weightGridWidth)

  lengthGridWidth = 0.001
  lengthSeq = seq(2,30,by=lengthGridWidth)

  # p(M|L,W) =~ P(L|M) * P(W|M) * P(M)
  maleProb = pWgivenM(W,280,weightSeq,weightGridWidth)*pLgivenM(L,12,lengthSeq
,lengthGridWidth)*pM

  # p(F|L,W) =~ P(L|F) * P(W|F) * P(F)
  womenProb = pWgivenF(W,300,weightSeq,weightGridWidth)*pLgivenF(L,14,lengthSe
q,lengthGridWidth)*pF

  # return normalized probabilities of classes
  list(male = maleProb/(maleProb+womenProb), female = womenProb/(maleProb+wome
nProb))

}

pred = naiveBayesClassifier(10,250)
print(pred)
```

```
$male
[1] 0.6336273

$female
[1] 0.3663727
```