Ec/ACM/CS 112. Problem Set 3

Ben Juarez

Question 1

preliminaries

```
set.seed(123) # initialize random seed
data = read.csv(file = "~/Desktop/data_task_duration_difficulty.csv")
```

build model objects

```
# Filter data
difficulty = data$difficulty
duration = data$duration

# Parameter grid
nGridPoints = 200
muGridMin = 0
muGridMax = 20
sigGridMin = 0.05
sigGridMax = 10
muGrid = seq(muGridMin, muGridMax, length.out = nGridPoints)
sigGrid = seq(sigGridMin, sigGridMax, length.out = nGridPoints)
muGridSize = (muGridMax - muGridMin) / nGridPoints
sigGridSize = (sigGridMax - sigGridMin) / nGridPoints

# Prior matrix
buildPriors = function() {
  priorM = matrix(rep(0, nGridPoints ^ 2),
                  nrow = nGridPoints,
                  ncol = nGridPoints,
                  byrow = TRUE)
  for (row in 1:nGridPoints) {
    for (col in 1:nGridPoints) {
      priorM[row,col] = muPrior[row] * sigPrior[col]
    }
  }
  priorM = priorM / sum(priorM)
  priorM = priorM / (muGridSize & sigGridSize)
  return(priorM)
}

# Compute posterior
computePost = function(data, priorM) {
```

```
  postM = matrix(rep(-1, nGridPoints ^ 2),
                 nrow = nGridPoints,
                 ncol = nGridPoints,
                 byrow = TRUE)
  for (row in 1:nGridPoints) {
    for (col in 1:nGridPoints) {
      muVal = muGrid[row]
      sigVal = sigGrid[col]
      loglike = sum(log(dnorm(data, muVal, sigVal)))
      postM[row,col] = exp(loglike) * priorM[row,col]
    }
  }
  postM = postM / (sum(postM) * muGridSize * sigGridSize)
  return(postM)
}
```

step 1: compute joint posterior

```
# Priors for mu
muMin = 0
muMax = 20
muPrior = dunif(muGrid, muMin, muMax)
muPrior = muPrior / (sum(muPrior) * muGridSize)

# Priors for sigma
sigMin = 0
sigMax = 10
sigPrior = dunif(sigGrid, sigMin, sigMax)
sigPrior = sigPrior / (sum(sigPrior) * sigGridSize)

priorM = buildPriors()
postM = computePost(duration, priorM)
```

step 2: compute marginal posterior distributions

```
margMu = rowSums(postM * sigGridSize)
margSig = colSums(postM * muGridSize)
```

step 3: compute summary statistics of marginal posteriors

```
muMean = sum(muGrid * margMu * muGridSize)
sigMean = sum(sigGrid * margSig * sigGridSize)
muSD = sqrt(sum(margMu * muGridSize * (muGrid - muMean)^2))
sigSD = sqrt(sum(margSig * sigGridSize * (sigGrid - sigMean)^2))

sum = 0
for (i in 1:nGridPoints) {
  for (j in 1:nGridPoints) {
    sum = sum + (muGrid[i] * muGridSize * sigGrid[j] * sigGridSize * postM[i,j])
  }
}
cv = sum - (muMean * sigMean)
```

```
paste("Mean of marginal posterior for mu: ", round(muMean,3))
```

```
## [1] "Mean of marginal posterior for mu:  7.2"
```

```
paste("Standard deviation of marginal posterior for mu: ", round(muSD,3))
```

```
## [1] "Standard deviation of marginal posterior for mu:  0.415"
```

```
paste("Mean of marginal posterior for sigma: ", round(sigMean,3))
```

```
## [1] "Mean of marginal posterior for sigma:  3.334"
```

```
paste("Standard deviation of marginal posterior for sigma: ", round(sigSD,3))
```

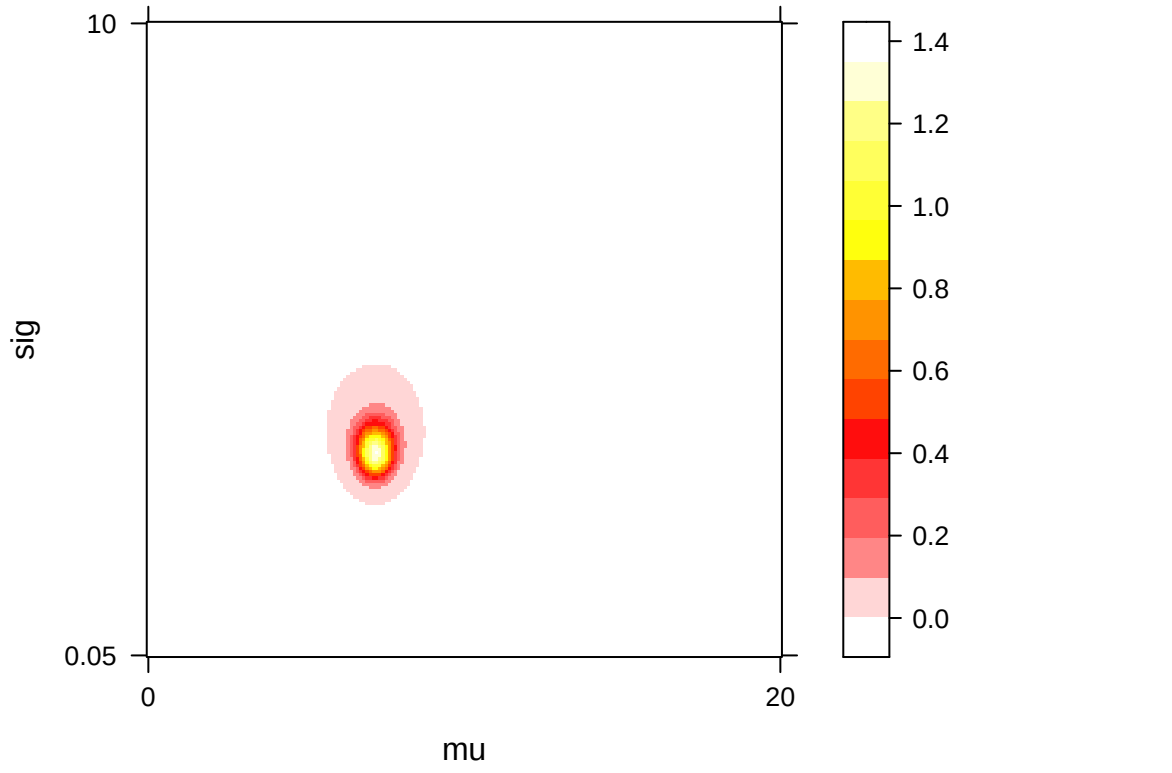
```
## [1] "Standard deviation of marginal posterior for sigma:  0.302"
```

```
paste("Covariance of mu and sigma: ", cv)
```

```
## [1] "Covariance of mu and sigma:  -2.98427949019242e-13"
```

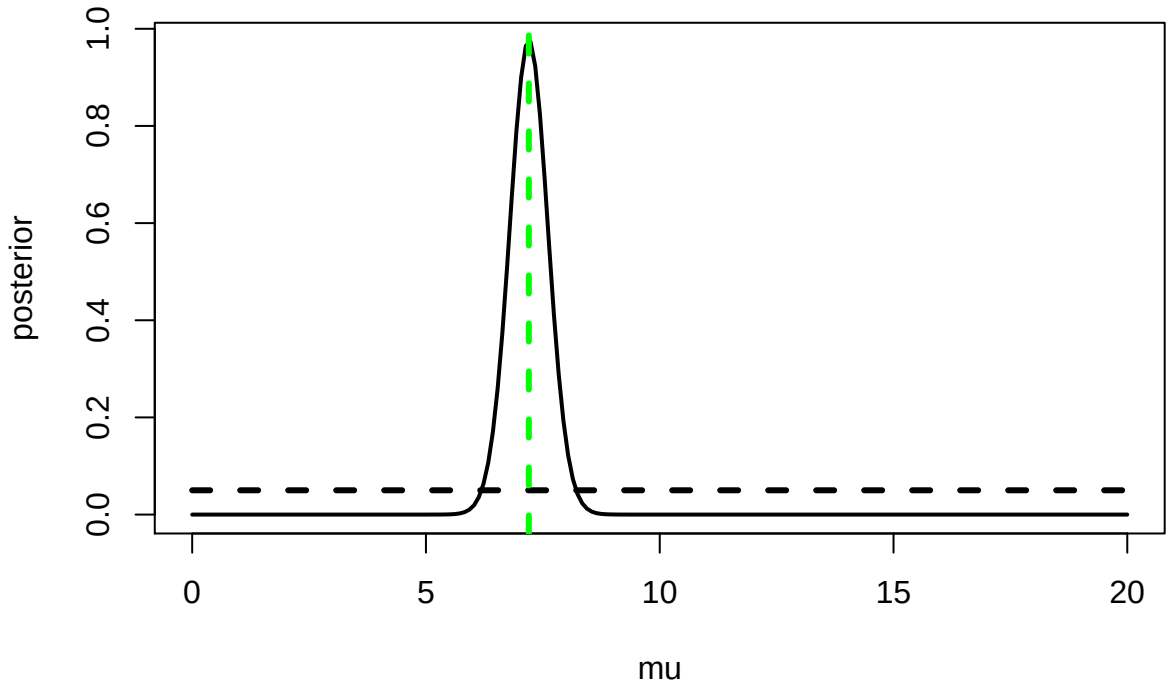
step 4: plot heat map of joint posterior

```
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "mu", ylab = "sig",
          scales=list(x=list(at=c(1,nGridPoints),labels=c(muGridMin,muGridMax)),
                      y=list(at=c(1,nGridPoints),labels=c(sigGridMin,sigGridMax))))
```

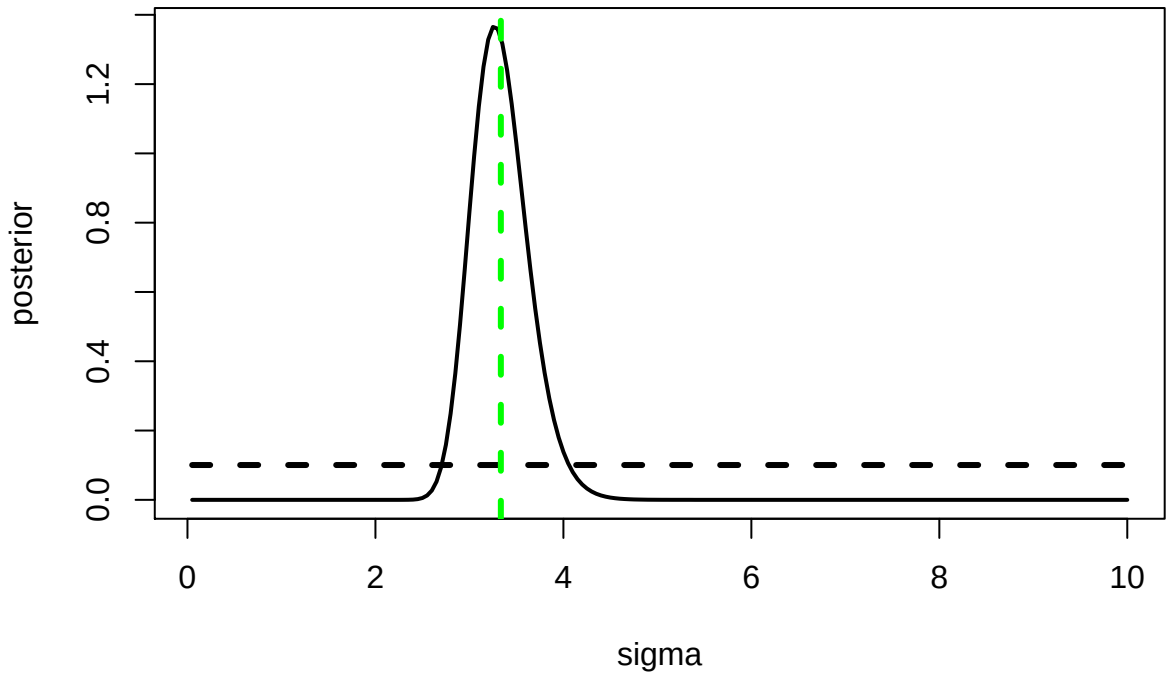


step 5: plot marginal posteriors

```
# visualize marginal posterior for mu
plot(muGrid, margMu, type="l", lwd=2, xlab="mu", ylab="posterior")
points(muGrid, muPrior, type="l", lwd=3, lty=2)
abline(v = muMean, lwd=3, lty=2, col = "green")
```




```
# visualize marginal posterior for sigma
plot(sigGrid, margSig, type="l", lwd=2, xlab="sigma", ylab="posterior")
points(sigGrid, sigPrior, type="l", lwd=3, lty=2)
abline(v = sigMean, lwd=3, lty=2, col = "green")
```



step 6: compute posterior prob mu < 5

```
prob = muGridSize * sum(margMu[1:50])
paste("P(mu < 5|data) =", prob)
```

```
## [1] "P(mu < 5|data) = 4.3688905160503e-07"
```

Question 2

preliminaries

```
set.seed(123)
data = read.csv(file = "~/Desktop/data_task_duration_difficulty.csv")
data = subset(data, !is.na(data$difficulty))
difficulty = data$difficulty # x
duration = data$duration # y

boundBetaGrid = 10
sizeBetaGrid = 0.1
lowerBoundSigmaGrid = 0.05
upperBoundSigmaGrid = 5
sizeSigmaGrid = 0.05

beta0Grid = seq(-boundBetaGrid,boundBetaGrid, by = sizeBetaGrid)
beta1Grid = seq(-boundBetaGrid,boundBetaGrid, by = sizeBetaGrid)
sigmaGrid = seq(lowerBoundSigmaGrid,upperBoundSigmaGrid, by = sizeSigmaGrid)
nBeta0Grid = length(beta0Grid)
nBeta1Grid = length(beta1Grid)
nSigmaGrid = length(sigmaGrid)
```

build model objects

```
likelihood = function(y,x, b0L, b1L, sL){
  loglike = sum(log(dnorm(y-b0L-b1L*x, mean = 0, sd = sL)))
  like = exp(loglike)
  return(like)
}

compPost = function(y, x){
  post = array(rep(-1, nBeta0Grid * nBeta1Grid * nSigmaGrid ),
               dim = c(nBeta0Grid, nBeta1Grid, nSigmaGrid ))
  for (nBeta0 in 1:nBeta0Grid) {
    b0 = beta0Grid[nBeta0]
    for (nBeta1 in 1:nBeta1Grid) {
      b1 = beta1Grid[nBeta1]
      for (nSigma in 1:nSigmaGrid) {
        s = sigmaGrid[nSigma]
        post[nBeta0,nBeta1,nSigma] = likelihood(y,x, b0, b1, s) * 1
      }
    }
  }
  post = post / (sum(post) * sizeBetaGrid^2 * sizeSigmaGrid)
  return(post)
```

```
}
```

Step 1: compute joint posterior

```
postFinal = array(rep(-1, 1 * nBeta0Grid * nBeta1Grid * nSigmaGrid),
                  dim = c(1, nBeta0Grid, nBeta1Grid, nSigmaGrid))
postFinal[1,,,] = compPost(duration, difficulty)
```

Step 2: compute marginal posteriors

```
margPostBeta0 = apply(postFinal,c(1,2),sum)[1,]/nBeta0Grid
margPostBeta1 = apply(postFinal,c(1,3),sum)[1,]/nBeta1Grid
margPostSigma = apply(postFinal,c(1,4),sum)[1,]/nSigmaGrid
```

Step 3: compute summary statistics of marginal posteriors

```
beta0Mean = sum(beta0Grid * margPostBeta0 * sizeBetaGrid)
beta1Mean = sum(beta1Grid * margPostBeta1 * sizeBetaGrid)
sigmaMean = sum(sigmaGrid * margPostSigma * sizeSigmaGrid)

beta0SD = sqrt(sum(margPostBeta0 * sizeBetaGrid * (beta0Grid - beta0Mean)^2))
beta1SD = sqrt(sum(margPostBeta1 * sizeBetaGrid * (beta1Grid - beta1Mean)^2))
sigmaSD = sqrt(sum(margPostSigma * sizeSigmaGrid * (sigmaGrid - sigmaMean)^2))

jointPost = apply(postFinal[1,,,],c(1,2),sum)
jointPost = jointPost / (sum(jointPost) * sizeBetaGrid^2)
sum = 0
for (i in 1:nBeta0Grid) {
  for (j in 1:nBeta1Grid) {
    sum = sum + (beta0Grid[i] * beta1Grid[j] * sizeBetaGrid^2 * jointPost[i,j])
  }
}
cv = sum - (beta0Mean * beta1Mean)

paste("Mean of marginal posterior for beta0: ", round(beta0Mean,3))
```

```
## [1] "Mean of marginal posterior for beta0:  2.865"
```

```
paste("Standard deviation of marginal posterior for beta0: ", round(beta0SD,3))
```

```
## [1] "Standard deviation of marginal posterior for beta0:  1.474"
```

```
paste("Mean of marginal posterior for beta1: ", round(beta1Mean,3))
```

```
## [1] "Mean of marginal posterior for beta1:  1.469"
```

```
paste("Standard deviation of marginal posterior for beta1: ", round(beta1SD,3))
```

```
## [1] "Standard deviation of marginal posterior for beta1:  0.474"
```

```
paste("Mean of marginal posterior for sigma: ", round(sigmaMean,3))
```

```
## [1] "Mean of marginal posterior for sigma:  3.1"
```

```
paste("Standard deviation of marginal posterior for sigma: ", round(sigmaSD,3))
```

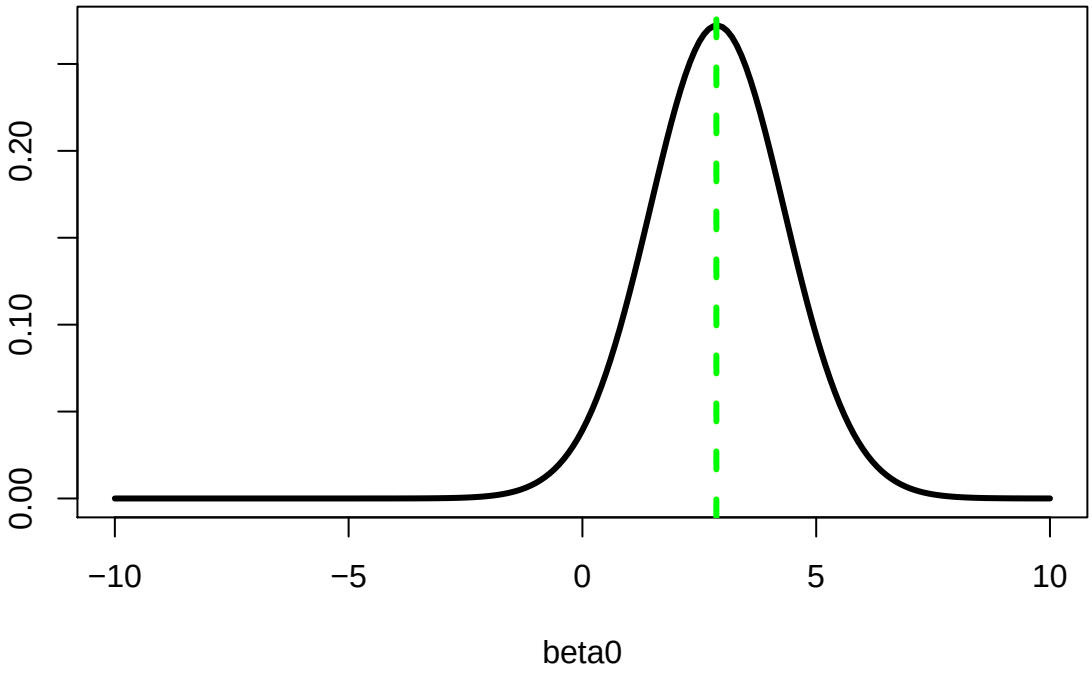
```
## [1] "Standard deviation of marginal posterior for sigma:  0.288"
```

```
paste("Covariance of beta0 and beta1: ", round(cv,3))
```

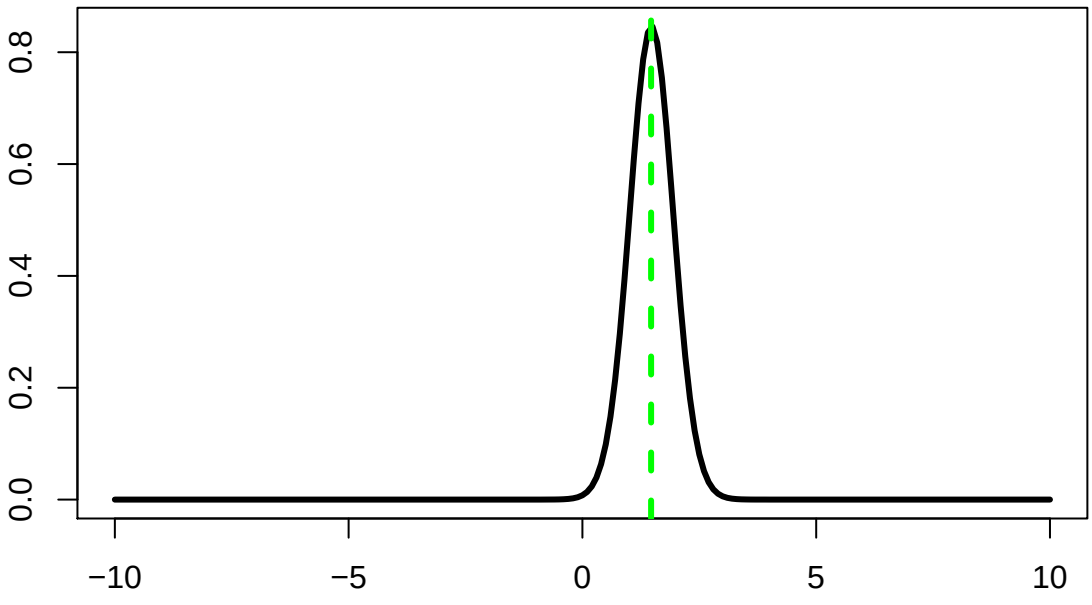
```
## [1] "Covariance of beta0 and beta1:  -0.634"
```

Step 4: plot marginal posterior densities

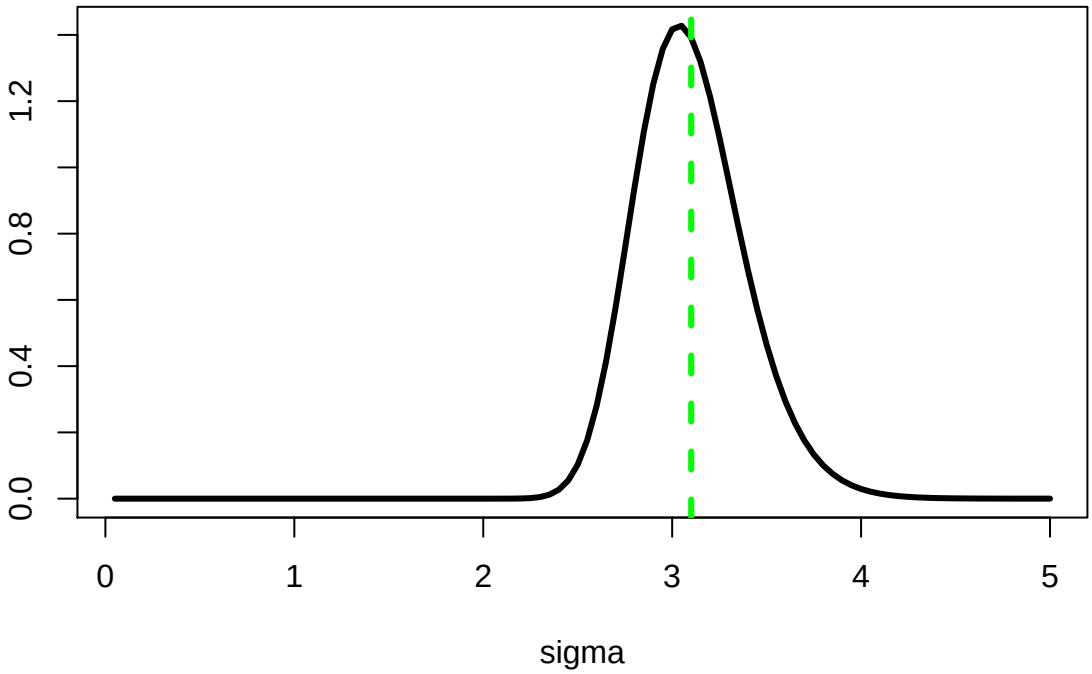
```
plot(beta0Grid, margPostBeta0, xlab = "beta0", ylab="", type = "l", lwd = 3)
abline(v = beta0Mean, lwd=3, lty=2, col = "green")
```




```
plot(beta1Grid, margPostBeta1, xlab = "beta1", ylab="", type = "l", lwd = 3)
abline(v = beta1Mean, lwd=3, lty=2, col = "green")
```



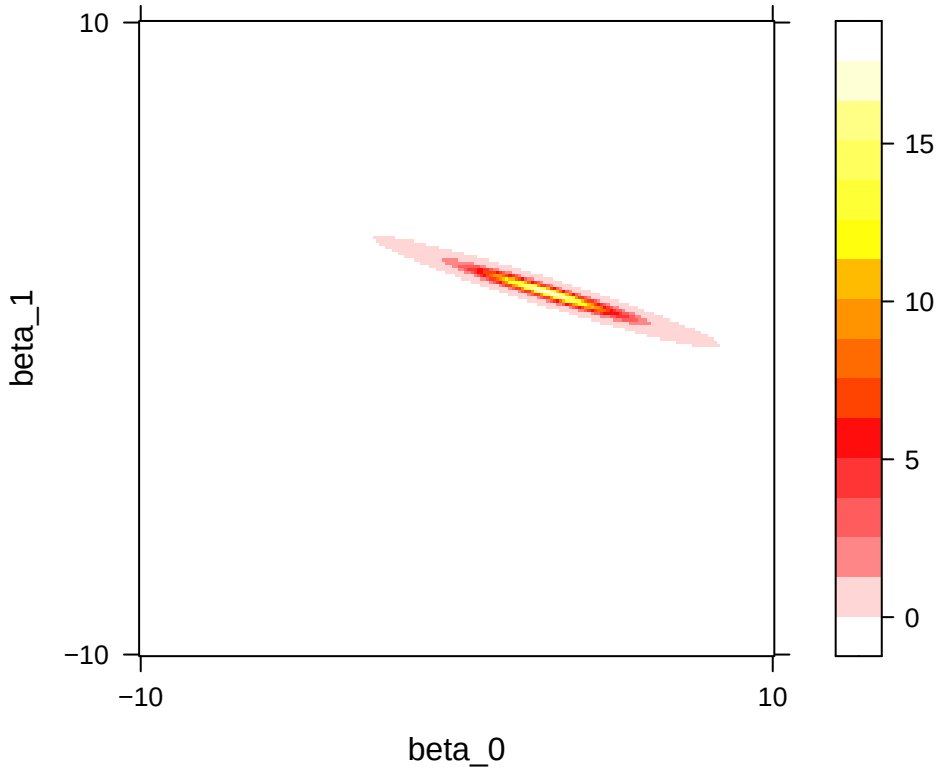

```
plot(sigmaGrid, margPostSigma, xlab = "sigma", ylab="", type = "l", lwd = 3)
abline(v = sigmaMean, lwd=3, lty=2, col = "green")
```



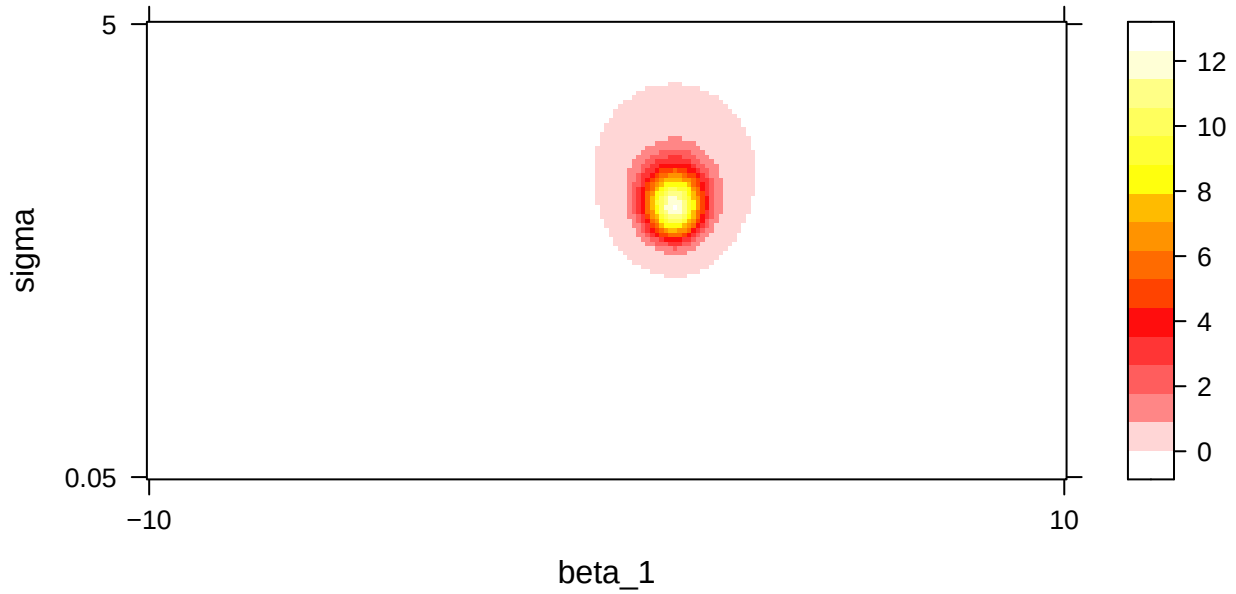
Step 5: Heat maps of joint posterior distributions

```
# joint posterior: beta0-beta1
library(lattice)
postLocal = postFinal[1,,,]
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
jointPost = apply(postLocal,c(1,2),sum)
```

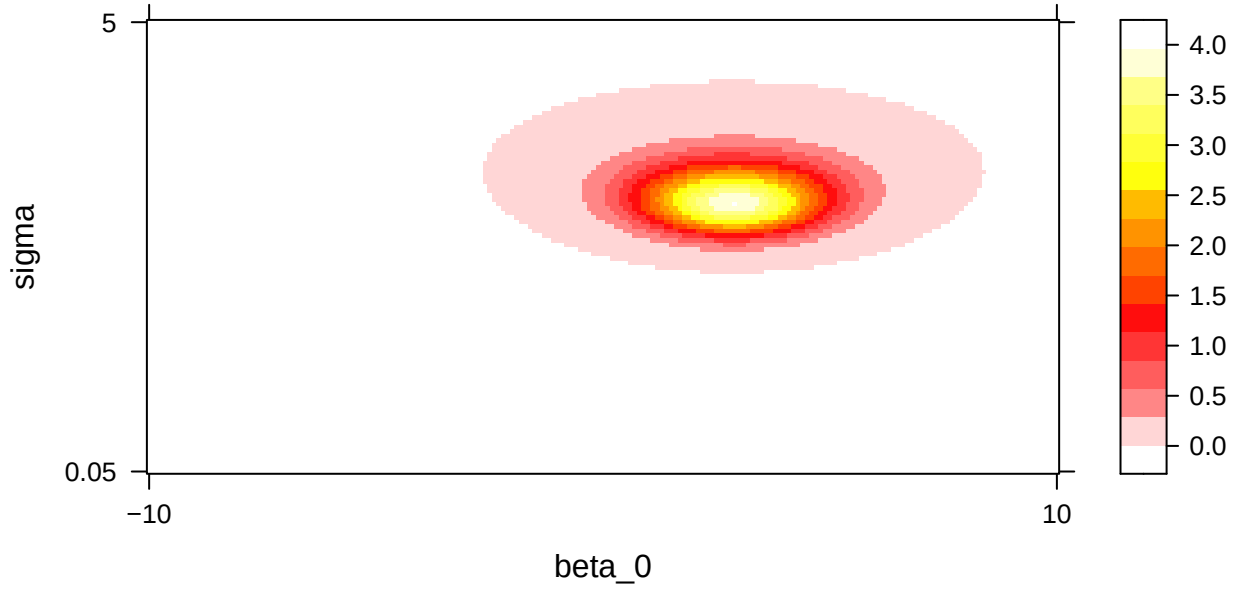
```
levelplot(jointPost, col.regions=new.palette(20), xlab = "beta_0", ylab = "beta_1",
          scales=list(x=list(at=c(1,nBeta0Grid), labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nBeta1Grid), labels=c(-boundBetaGrid,boundBetaGrid))))
```




```
# joint posterior: beta1-sigma
library(lattice)
postLocal = postFinal[1,,,]
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
jointPost = apply(postLocal,c(2,3),sum)
levelplot(jointPost, col.regions=new.palette(20), xlab = "beta_1", ylab = "sigma",
          scales=list(x=list(at=c(1,nBeta1Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nSigmaGrid),
                             labels=c(lowerBoundSigmaGrid,upperBoundSigmaGrid))))
```



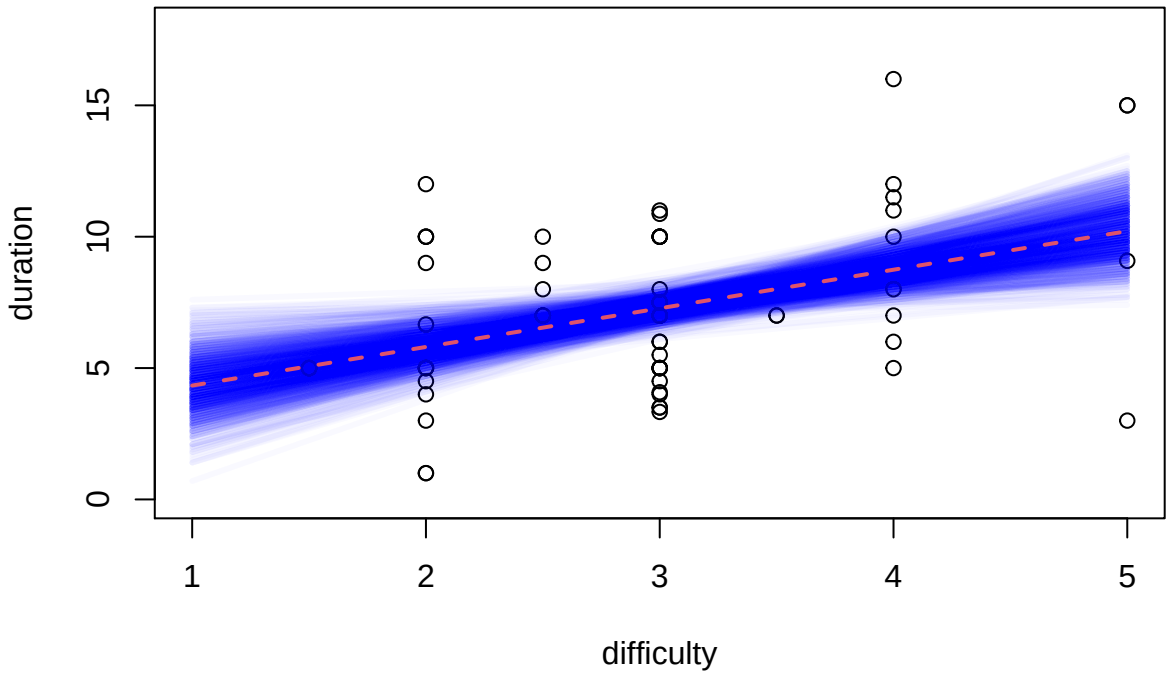

```
# joint posterior: beta0-sigma
library(lattice)
postLocal = postFinal[1,,,]
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
jointPost = apply(postLocal,c(1,3),sum)
levelplot(jointPost, col.regions=new.palette(20), xlab = "beta_0", ylab = "sigma",
          scales=list(x=list(at=c(1,nBeta0Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nSigmaGrid),
                             labels=c(lowerBoundSigmaGrid,upperBoundSigmaGrid))))
```



Step 6: Visualize uncertainty in posterior regression lines

```
xGrid = seq(1,5, by=0.1)
margBeta0 = apply(postFinal[1,,,],c(1),sum)
```

```
margBeta1GivenBeta0 = array(rep(-1,nBeta0Grid * nBeta1Grid),
                            dim= c(nBeta0Grid, nBeta1Grid))
for (nBeta0 in 1:nBeta0Grid) {
  margBeta1GivenBeta0[nBeta0,] = apply(postFinal[1,nBeta0,,],c(1),sum)
}
plot(difficulty,duration,xlim=c(1,5),ylim=c(0,18))
for (sim in 1:1000) {
  b0Index = sample(1:nBeta0Grid, 1, prob=margBeta0)
  b1Index = sample(1:nBeta1Grid, 1, prob=margBeta1GivenBeta0[b0Index,])
  b0Sample = beta0Grid[b0Index]
  b1Sample = beta1Grid[b1Index]
  points(xGrid, b0Sample + b1Sample*xGrid, type="l",lwd=3,
         col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.025))
}
points(xGrid, beta0Mean + beta1Mean*xGrid, lwd=2, col=2, lty=2, type="l")
```



Question 3

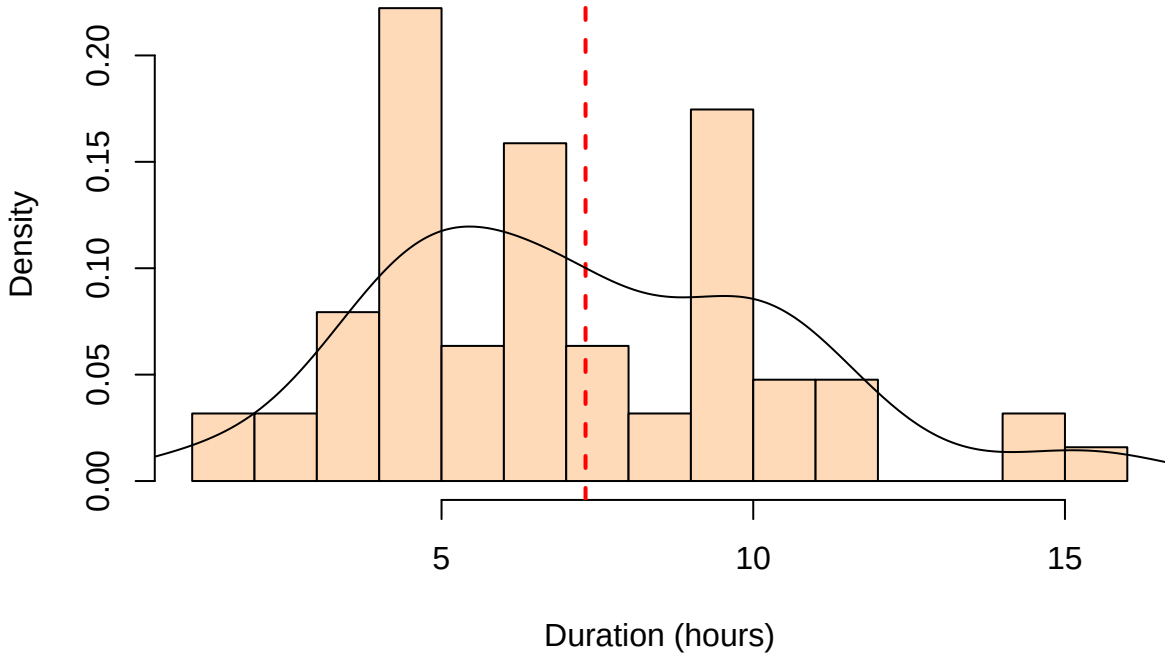
preliminaries

```
set.seed(123)
data = read.csv(file = "~/Desktop/data_task_duration_difficulty.csv")
data = subset(data, !is.na(data$difficulty))
difficulty = data$difficulty # x
duration = data$duration # y
```

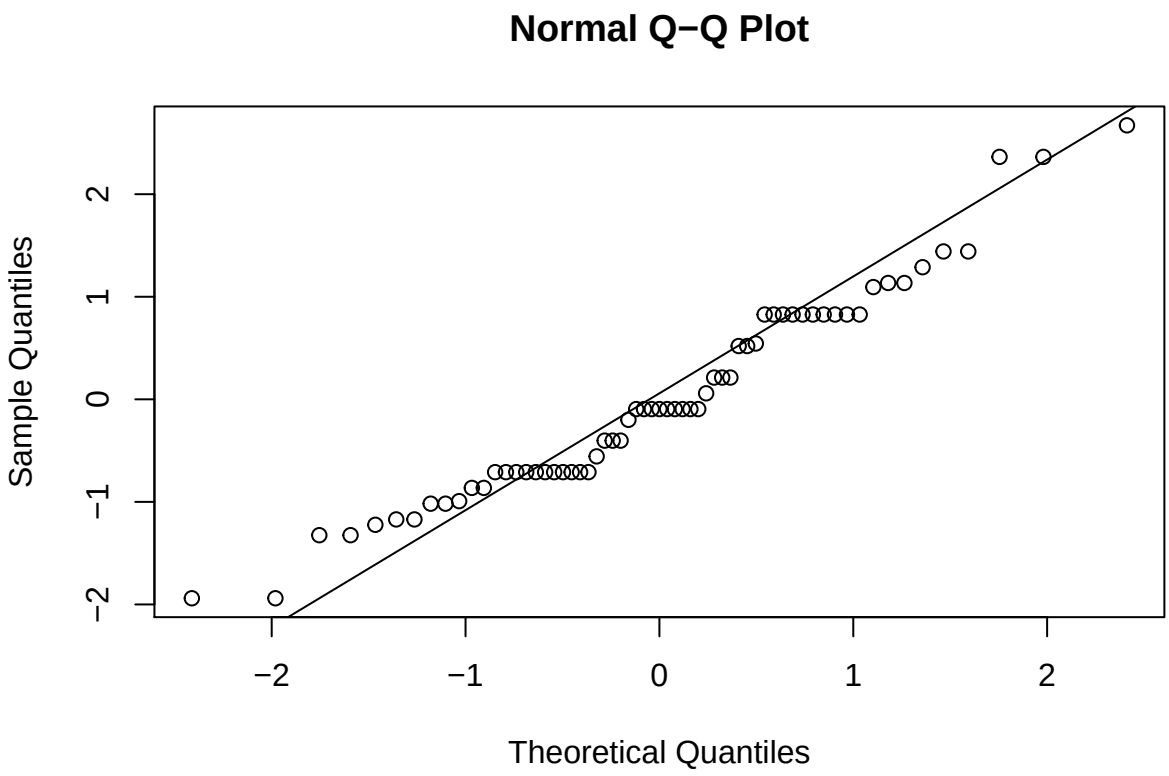
Step 1: About normality of duration data

```
hist(duration,
     breaks = 15,
```

```
     col="peachpuff",
     border="black",
     prob = TRUE,
     xlab = "Duration (hours)",
     main = "")
abline(v=mean(duration), lty=2, lwd=2, col="red")
lines(density((duration)))
```




```
standardized_y = (duration - mean(duration))/sd(duration)
qqnorm(standardized_y)
qqline(standardized_y)
```



It seems that these plots follow the normality assumption decently well. The density line in the histogram resembles a bell curve and the points on the normal q-q plot are following the straight line pretty well.

Step 2: About normality of errors in regression model

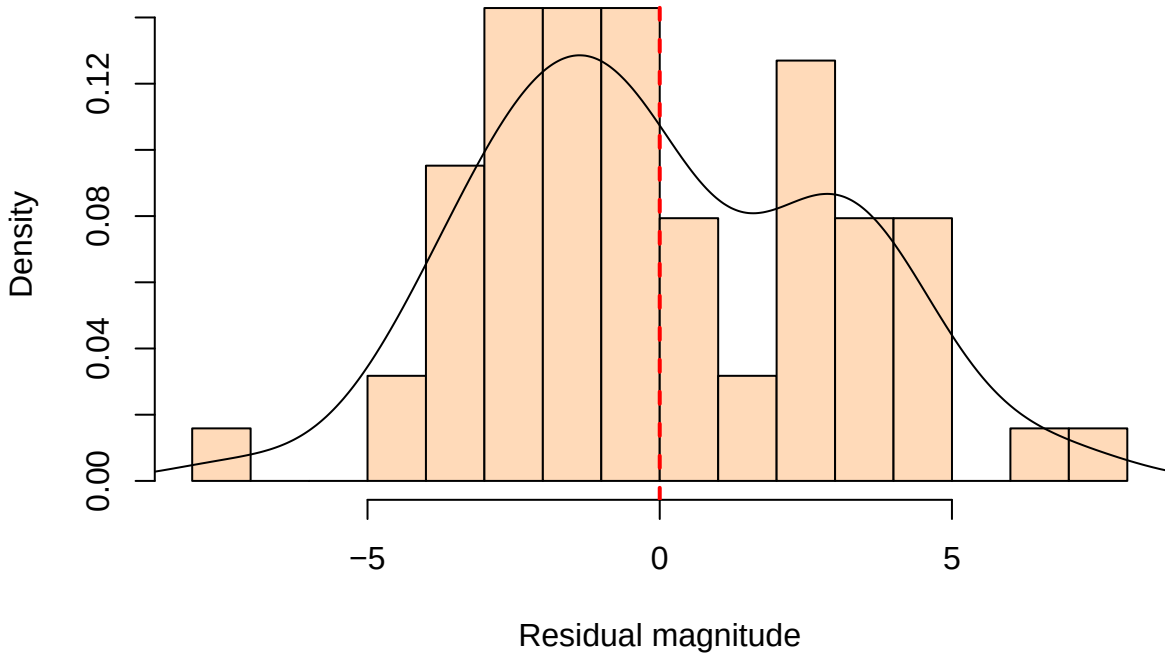
```
duration_lm = lm(duration ~ difficulty, data = data)
residuals = resid(duration_lm)

paste("Beta 0 estimate:", round(coef(duration_lm)[1],3))
```

```
## [1] "Beta 0 estimate: 2.88"
```

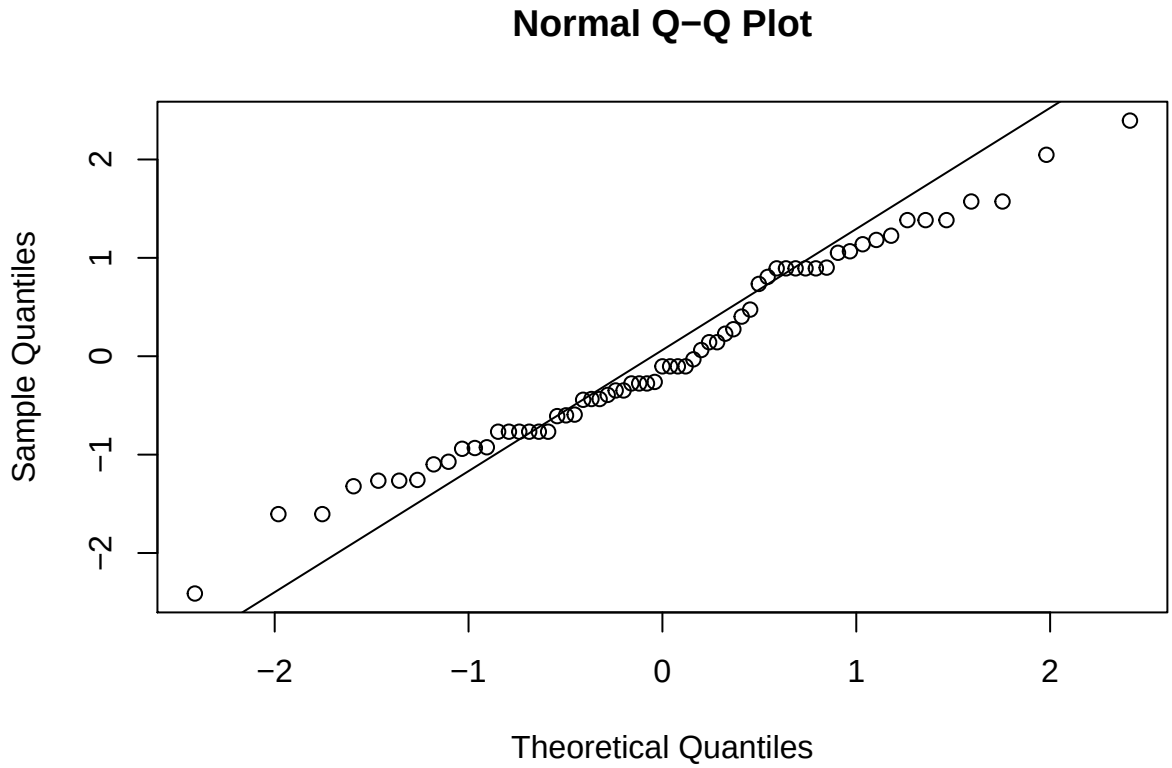
```
paste("Beta 1 estimate:", round(coef(duration_lm)[2],3))
```

```
## [1] "Beta 1 estimate: 1.477"
```

```
hist(residuals,
     breaks = 20,
     col="peachpuff",
     border="black",
     prob = TRUE,
     xlab = "Residual magnitude",
     main = "")
abline(v=mean(residuals), lty=2, lwd=2, col="red")
lines(density((residuals)))
```




```
standardized_y = (residuals - mean(residuals))/sd(residuals)
qqnorm(standardized_y)
qqline(standardized_y)
```



It seems like these plots are also consistent with the normality assumption since the histogram density line has a good general bell curve shape and the points on the normal q-q plot are also linearly distributed well for the most part.