STA_C67 A1

Question 1

a) Let's use the rnorm() function to get a sample, then calculate the mean and standard dev from there.

```
library(rcompanion)
set.seed(1006274274);

sampleSize <- 10;
nums<-rnorm(n=sampleSize, mean = 2,sd=5);


sampleMean <- mean(nums);

standardError <- sd(nums) / sqrt(sampleSize);

cat("The sample mean is:",sampleMean,"\n");
```

```
## The sample mean is: 3.449081
```

```
cat("The standard error is:", standardError,"\n");
```

```
## The standard error is: 1.25926
```

b)

```
linear_model <- lm( nums~ 1);

lin_model_intercept <- coef(linear_model)["(Intercept)"];

lin_model_std_err <- coef(summary(linear_model))[, "Std. Error"];

cat("The intercept (which is just the mean) is:",lin_model_intercept, "\n" );
```

```
## The intercept (which is just the mean) is: 3.449081
```

```
cat("The standard error is:",lin_model_std_err, "\n" );
```

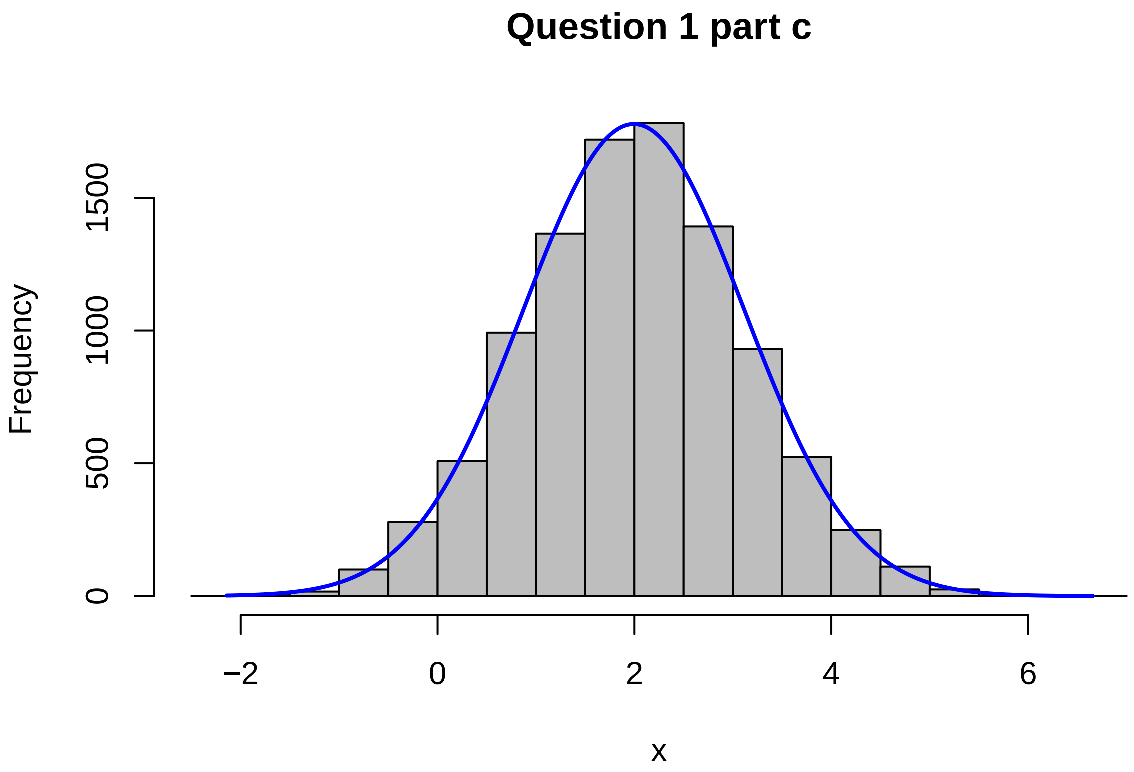
```
## The standard error is: 1.25926
```

These are identical to the results in part (a). This is because we've created a linear model with `lm(nums~1)`, which is equivalent to `nums[i] = B0(1) + X1(0) + e[i]`, . If we look at the mean of nums, we find that `mean(nums) == E(nums[i]) == E(B0 (1) + X1(0) + e[i]) == E(B0) + E(e[i])) == B0 + 0 == B0`. Thus the intercept is just the mean of the sample data. Similarly, the variance of nums is just σ^2, which is just the variance of the residuals, and thus the same logic goes with standard error, which is just $\sigma/sqrt(n)$.

c) This took a lot of googling.

```
meanList <- list();
for(i in 1:10000){
  randomSample <-rnorm(n=20, mean = 2,sd=5);
  currSampleMean = mean(randomSample);
  meanList <- append(meanList,currSampleMean);
}

plotNormalHistogram( unlist(meanList), prob = FALSE,
                     main = "Question 1 part c",
                     length = 1000)
```



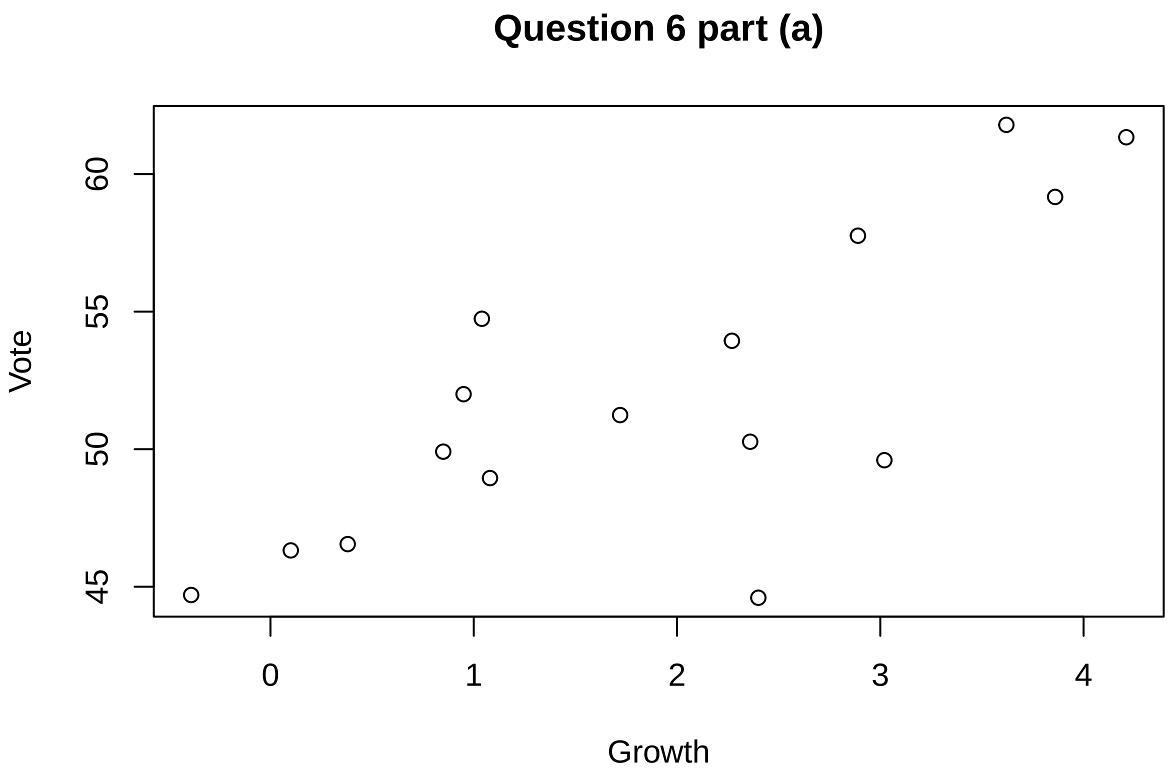
Question 6:

a)

```
myTable <- read.table("vote.txt", header = TRUE, sep = "", dec = ".")

growthArr <-myTable$growth;
voteArr <- myTable$vote;

# The growth is independent var, vote percentage is dependent var
plot(growthArr,voteArr,main="Question 6 part (a)", xlab="Growth", ylab="Vote")
```



b)

```
lin_model <- lm(voteArr ~ growthArr);
intercept <- coef(lin_model)["(Intercept)"]
slope <- coef(lin_model)["growthArr"]

cat("The intercept (B0 hat) is:", intercept,"\n");
```

```
## The intercept (B0 hat) is: 46.24765
```

```
cat("The slope (B1 hat) is:", slope,"\n");
```

```
## The slope (B1 hat) is: 3.060528
```

That means that if there is no economic growth, the incumbent party can expect to have a vote percentage of $\hat{B}_0$ It also means that for every 1% of economic growth, the incumbent party can expect to gain 3% in votes.

c) There's probably a more efficient way of coding this in R, but I can't be bothered to learn R properly.

```
rowNumOfYear2008 <- match(2008,myTable$year);
growthInYear2008 <- growthArr[rowNumOfYear2008]

newDataFramePredictor <- data.frame(growthArr=growthInYear2008);

votePercentageOfIncumbent <- predict(lin_model,newDataFramePredictor);

incumbentOrOppositionStr <-if(votePercentageOfIncumbent >= 50) "Incumbent" else "Opposition";

cat("The vote % of incumbent party is predicted to be:", votePercentageOfIncumbent,".\n So the",
    incumbentOrOppositionStr, "party is expected to win.\n That party is led by: ",
    if (incumbentOrOppositionStr == "Incumbent")
            myTable$inc_party_candidate[rowNumOfYear2008]
    else myTable$other_candidate[rowNumOfYear2008], "\n" );
```

```
## The vote % of incumbent party is predicted to be: 46.5537 .
##  So the Opposition party is expected to win.
##  That party is led by:  Obama
```

If we calculate this by hand, we can model it as $Y = \beta_0 + \beta_1 X$, and plugging in the numbers, we get $Y = 46.55$, which is still less than 50%. So Obama (the opposition), is still expected to win. Note that this assumes that every citizen will vote, which is obviously not true in practice.

d) Let's use the pearson least squares method:

```r
cor(growthArr, voteArr,  method = "pearson");
```

```
## [1] 0.7614763
```

There is a pretty strong positive correlation between growth and vote share.

e) We can first calculate this by hand, by using the formula $\hat{\beta}_j \pm t_{14}^{0.95} SE(\hat{\beta}_j)$. For the t value, we get 2.145, and we have the standard error $SE(\hat{B}_j) = 0.6963$ from before by calling `summary(lin_model)`, and we know that $\hat{B}_j = 3.0605$. From those calculations, we get a 95% confidence interval of [1.567,4.554].

We can also calculate this with R:

```r
confint(lin_model, 'growthArr', level=0.95)
```

```
##               2.5 %   97.5 %
## growthArr 1.567169 4.553887
```

f) I first calculated this by hand, then figured out how to calculate it in R. We should notice that the expression can be turned into: $P(\hat{\beta}_1 - \beta > 1) = P(\frac{-1}{SE(\hat{\beta}_1)} < \frac{\hat{\beta}_1 - \beta}{SE(\hat{\beta}_1)} < \frac{1}{SE(\hat{\beta}_1)})$. Notice that the middle term is just our t-statistic, which we can use to calculate our probability. The following code does just that:

```r
#Prob a better way of writing this
b1HatStdError <- summary(lin_model)$coefficients[, 2]["growthArr"];

degFreedom <- length(growthArr) - 2;

t_value <- 1/b1HatStdError;

#Multiply by 2, since we need to account for both sides.
#We can also multiply by two since T-distribution is symmetric.
probabilityRes <- (1- pt(t_value, df = degFreedom)) * 2;

cat("The probability calculation gives us:", probabilityRes,"\n");
```

```
## The probability calculation gives us: 0.1729088
```