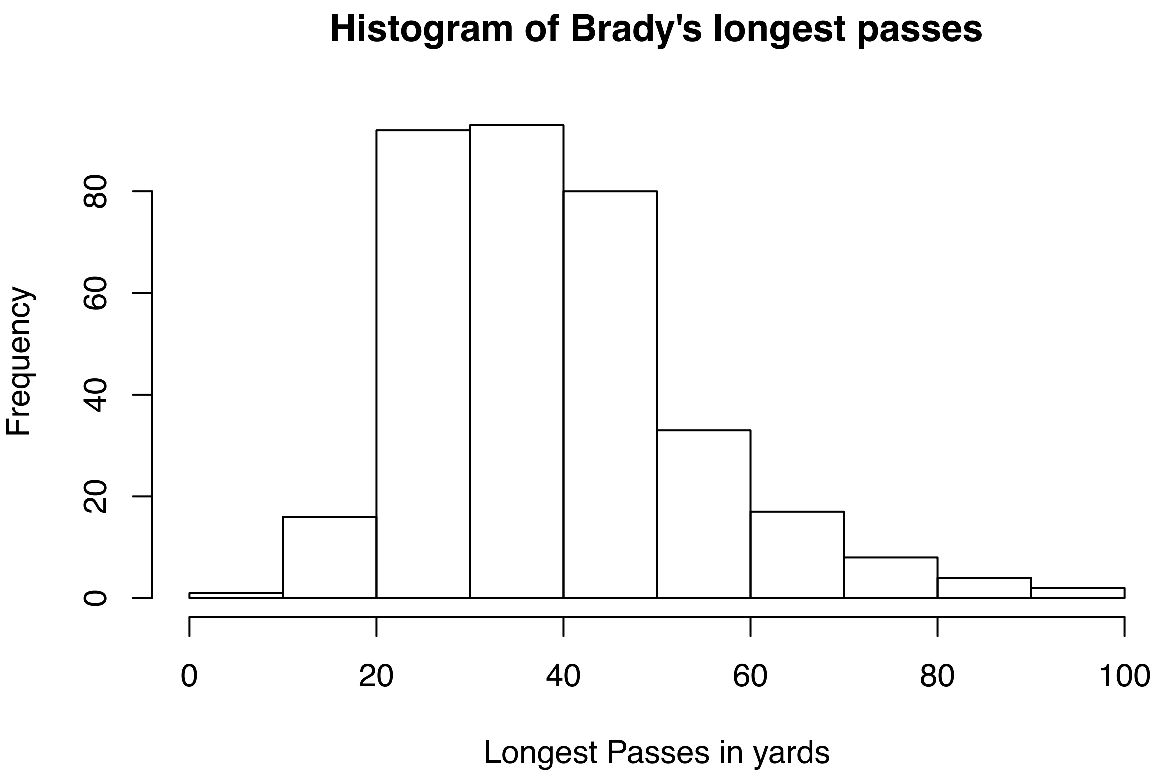
STAT341 A3 Markdown

QUESTION 1: Parameter Estimation in the Gamma Distribution

(a) Construct a histogram of $y =$ `LNG`, Tom Brady's longest passes in each game. Be sure to include an informative title and axis labels. Comment on the suitability of the gamma distribution as a potential model for this data.

```
brady <- read.csv("brady.csv")

hist(brady$LNG, main = "Histogram of Brady's longest passes",
     xlab = "Longest Passes in yards")
```



Based on the histogram, it looks that it is right-skewed which suggests that the gamma distribution is potentially an accurate model for this data.

(b) Derive the MM estimates of α and β. You may use without proof or derivation the fact that $\text{E}[Y] = \alpha\beta$ and $\text{Var}[Y] = \alpha\beta^2$, if $Y \sim \text{GAM}(\alpha, \beta)$. Using these expressions and R, calculate the MM estimates of α and β for $y =$ `LNG`.

```r
# calculate E[Y] and E[Y^2] for y=LNG
ey <- mean(brady$LNG)
ey2 <- (sum(brady$LNG))^2 / 346
```

Based on the calculation from R:

$$\mathrm{E}[Y] = 39.3815 \qquad \mathrm{E}[Y^2] = 536612.4$$

and so:

$$\mathrm{Var}[Y] = 536612.4 - (39.3815)^2 = 535061.5$$

Now we use $Y \sim \mathrm{GAM}(\alpha, \beta)$ and solve for α and β

$$39.3815 = \alpha\beta \qquad 535061.5 = \alpha\beta^2$$

Solving this system of equations gives:

$$\alpha = 0.00289855 \qquad \beta = 13586.62$$

(c) Derive the log-likelihood function $l(\alpha, \beta; \mathcal{P})$

$$l(\alpha, \beta; \mathcal{P}) = \ln([\Gamma(\alpha)\beta^{\alpha}]^{-N} \times \left[\prod_{i=1}^{N} y_i\right]^{\alpha-1} \times \exp\left\{-\frac{1}{\beta}\sum_{i=1}^{N} y_i\right\}).$$

$$l(\alpha, \beta; \mathcal{P}) = -N\left[\ln(\Gamma(\alpha)) + \alpha\ln\beta\right] + \left[(\alpha-1)\sum_{i=1}^{N}\ln(y_i)\right] - \left(\frac{1}{\beta}\sum_{i=1}^{N} y_i\right)$$

(d) Determine the partial derivatives $\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\alpha}$ and $\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\beta}$. Note that derivatives of $\Gamma(\alpha)$ with respect to α may simply be written as $\Gamma'(\alpha)$.

$$\frac{\partial l(\alpha, \beta; \mathcal{P})}{\partial\alpha} = -N\left[\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \ln\beta\right] + \left[\sum_{i=1}^{N}\ln(y_i)\right]$$

$$\frac{\partial l(\alpha, \beta; \mathcal{P})}{\partial\beta} = -N\left[\frac{\alpha}{\beta}\right] + \left(\frac{1}{\beta^2}\sum_{i=1}^{N} y_i\right)$$

(e) Determine the second partial derivatives $\frac{\partial^2 l(\alpha,\beta;\mathcal{P})}{\partial\alpha^2}$, $\frac{\partial^2 l(\alpha,\beta;\mathcal{P})}{\partial\alpha\beta}$, $\frac{\partial^2 l(\alpha,\beta;\mathcal{P})}{\partial\beta\alpha}$ and $\frac{\partial^2 l(\alpha,\beta;\mathcal{P})}{\partial\beta^2}$. Note that derivatives of $\Gamma'(\alpha)$ with respect to α may simply be written as $\Gamma''(\alpha)$.

$$\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\alpha^2} = -N\left[\frac{(\Gamma''(\alpha)\Gamma(\alpha)) - (\Gamma'(\alpha))^2}{(\Gamma(\alpha))^2}\right]$$

$$\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\alpha\beta} = -N\left[\frac{1}{\beta}\right]$$

$$\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\beta\alpha} = -N\left[\frac{1}{\beta}\right]$$

$$\frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\beta^2} = N\left[\frac{\alpha}{\beta^2}\right] - \left(\frac{2}{\beta^3}\sum_{i=1}^{N} y_i\right)$$

(f) Using the derivatives found in parts (d) and (e), define the vector $\boldsymbol{\psi}(\alpha,\beta;\mathcal{P})$ and the matrix $\boldsymbol{\psi}'(\alpha,\beta;\mathcal{P})$ required for the Newton-Raphson algorithm.

$$\boldsymbol{\psi}(\alpha,\beta;\mathcal{P}) = \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\alpha} \\ \frac{\partial l(\alpha,\beta;\mathcal{P})}{\partial\beta} \end{bmatrix} = \begin{bmatrix} -N\left[\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} + \ln\beta\right] + \left[\sum_{i=1}^{N}\ln(y_i)\right] \\ -N\left[\frac{\alpha}{\beta}\right] + \left(\frac{1}{\beta^2}\sum_{i=1}^{N} y_i\right) \end{bmatrix}$$

$$\boldsymbol{\psi}'(\alpha,\beta;\mathcal{P}) = \begin{bmatrix} \frac{\partial\psi_1(\alpha,\beta;\mathcal{P})}{\partial\alpha} & \frac{\partial\psi_1(\alpha,\beta;\mathcal{P})}{\partial\beta} \\ \frac{\partial\psi_2(\alpha,\beta;\mathcal{P})}{\partial\alpha} & \frac{\partial\psi_2(\alpha,\beta;\mathcal{P})}{\partial\beta} \end{bmatrix} = \begin{bmatrix} -N\left[\frac{(\Gamma''(\alpha)\Gamma(\alpha))-(\Gamma'(\alpha))^2}{(\Gamma(\alpha))^2}\right] & -N\left[\frac{1}{\beta}\right] \\ -N\left[\frac{1}{\beta}\right] & N\left[\frac{\alpha}{\beta^2}\right] - \left(\frac{2}{\beta^3}\sum_{i=1}^{N} y_i\right) \end{bmatrix}$$

(g) Write *factory functions* `createGammaPsiFn(y)` and `createGammaPsiPrimeFn(y)` which take in as input the data `y`, and which respectively return as output functions corresponding to $\boldsymbol{\psi}(\alpha,\beta;\mathcal{P})$ and $\boldsymbol{\psi}'(\alpha,\beta;\mathcal{P})$ determined in part (f).

```r
createGammaPsiFn <- function(y) {
  function(theta) {
    alpha <- theta[1]
    beta <- theta[2]
    c(-346*(digamma(alpha)+log(beta)) + sum(log(y)),
      -346*(alpha/beta) + (1/beta^2)*(sum(y)))
  }
}

createGammePsiPrimeFn <- function(y) {
  function(theta) {
    alpha <- theta[1]
    beta <- theta[2]
    val <- matrix(0, nrow = 2, ncol = 2)
    val[1,1] = -346*(trigamma(alpha))
    val[1,2] = -346*(1/beta)
    val[2,1] = -346*(1/beta)
    val[2,2] = -346*(alpha/beta^2) - (2/beta^3)*(sum(y))
    return(val)
  }
}
```

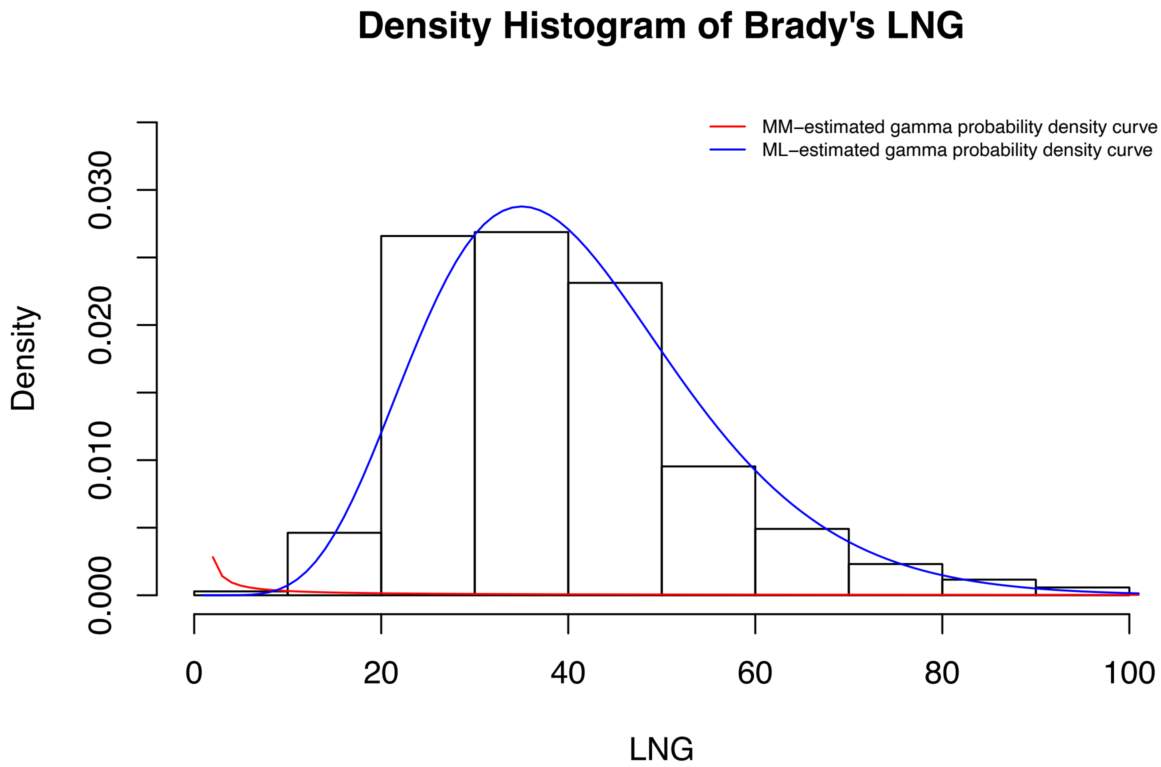
(h) Using the `NewtonRaphson` function from class, together with the `psiFn` and `psiPrimeFn` functions created by your factory functions from part (g), calculate $\widehat{\alpha}$ and $\widehat{\beta}$, the maximum likelihood estimates of α and β associated with $y =$ `LNG`. Start the algorithm at (α_0, β_0) where α_0 and β_0 are the MM estimates of α and β you calculated in part (b). For full points be sure to include the output from the `NewtonRaphson` function.

```
y <- brady$LNG
nr <- NewtonRaphson(theta = c(0.00289855, 13586.62),
                    psiFn = createGammaPsiFn(y), psiPrimeFn = createGammePsiPrimeFn(y))
print(nr)
```

```
## $theta
## [1] 7.185768 5.499906
##
## $converged
## [1] FALSE
##
## $iteration
## [1] 101
##
## $fnValue
## [1]  0.001088922 -1.596153887
```

(i) Construct a density histogram of $y =$ `LNG` and overlay the MM-estimated gamma probability density curve as well as the ML-estimated gamma probability density curve. Be sure to distinguish between these curves with a legend and include an informative title and axis labels.

```
# MM-estimated curve has (alpha = 0.00289855, beta = 13586.62)
# ML-estimated curve has (alpha = 7.185768, beta = 5.499906)
xlen <- seq(0,100)
hist(brady$LNG, prob = TRUE, main = "Density Histogram of Brady's LNG",
     xlab = "LNG", ylim=c(0, 0.035))
lines(dgamma(x=xlen, shape=0.00289855, scale = 13586.62), col = "red")
lines(dgamma(x=xlen, shape=7.185768, scale = 5.499906), col = "blue")
legend("topright", legend=c("MM-estimated gamma probability density curve",
                            "ML-estimated gamma probability density curve"),
       col=c("red","blue"), lty=c(1,1), bty="n",
       cex=0.6)
```



Density Histogram of Brady's LNG
MM−estimated gamma probability density curve
ML−estimated gamma probability density curve
Density
0.000
0.010
0.020
0.030
0
20
40
60
80
100
LNG

QUESTION 2: Fit All the Regressions with IRLS

(a) Using the `irls` function with `tolerance = 1e-10` and `maxIterations = 1000`, determine $\left(\widehat{\alpha}_{LS}, \widehat{\beta}_{LS}\right)$, the least squares estimates of α and β. Start the algorithm at $(\alpha_0, \beta_0) = (1, 1)$. For full points be sure to include the output from the `irls` function.

```
LSrhoPrime <- function(resid) {
  2 * resid
}

# irls function from class
IRLSresult <- irls(brady$YDS, brady$AVG, theta = c(1, 1), rhoPrimeFn = LSrhoPrime,
                   tolerance = 1e-10, maxIterations = 1000)
print(IRLSresult)
```

```
## $theta
## [1] 264.89306  28.43653
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 2
```

Based on the output, $\left(\widehat{\alpha}_{LS}, \widehat{\beta}_{LS}\right) = (264.89306, 28.43653)$

(b) Using the `irls` function with `tolerance = 1e-10` and `maxIterations = 1000`, determine $\left(\widehat{\alpha}_{LAD}, \widehat{\beta}_{LAD}\right)$, the least absolute deviations estimates of α and β. Start the algorithm at $(\alpha_0, \beta_0) = \left(\widehat{\alpha}_{LS}, \widehat{\beta}_{LS}\right)$. For full points be sure to include the output from the `irls` function.

```
LADrhoPrime <- function(resid) {
  sign(resid)
}

# irls function from class
IRLSresult2 <- irls(brady$YDS, brady$AVG, theta = c(264.89306, 28.43653), rhoPrimeFn
                    = LADrhoPrime, tolerance = 1e-10, maxIterations = 1000)
print(IRLSresult2)
```

```
## $theta
## [1] 264.89306  28.43653
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 2
```

Based on the output, $\left(\widehat{\alpha}_{LAD}, \widehat{\beta}_{LAD}\right) = (264.89306, 28.43653)$

(c) Using the `irls` function with `tolerance = 1e-10` and `maxIterations = 1000`, determine

$\left(\widehat{\alpha}_{Huber}, \widehat{\beta}_{Huber}\right)$, the Huber robust regression estimates of α and β. Start the algorithm at $(\alpha_0, \beta_0) = \left(\widehat{\alpha}_{LS}, \widehat{\beta}_{LS}\right)$. Use $k = 1.345$ in the Huber function

```
huber.fn.prime <- function(resid, k = 1.345) {
  val = resid
  subr = abs(resid) > k
  val[subr] = k * sign(resid[subr])
  return(val)
}

# irls function from class
IRLSresult3 <- irls(brady$YDS, brady$AVG, theta = c(264.89306, 28.43653), rhoPrimeFn = huber.fn.prime,
                    tolerance = 1e-10, maxIterations = 1000)
print(IRLSresult3)
```

```
## $theta
## [1] 264.89306  28.43653
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 2
```

Based on the output, $\left(\widehat{\alpha}_{Huber}, \widehat{\beta}_{Huber}\right) = (264.89306, 28.43653)$

(d) Using the `irls` function with `tolerance = 1e-10` and `maxIterations = 1000`, determine $\left(\widehat{\alpha}_{Tukey}, \widehat{\beta}_{Tukey}\right)$, the Tukey robust regression estimates of α and β. Start the algorithm at $(\alpha_0, \beta_0) = \left(\widehat{\alpha}_{LS}, \widehat{\beta}_{LS}\right)$. Use $k = 4.685$ in the Tukey bisquare (biweight) function

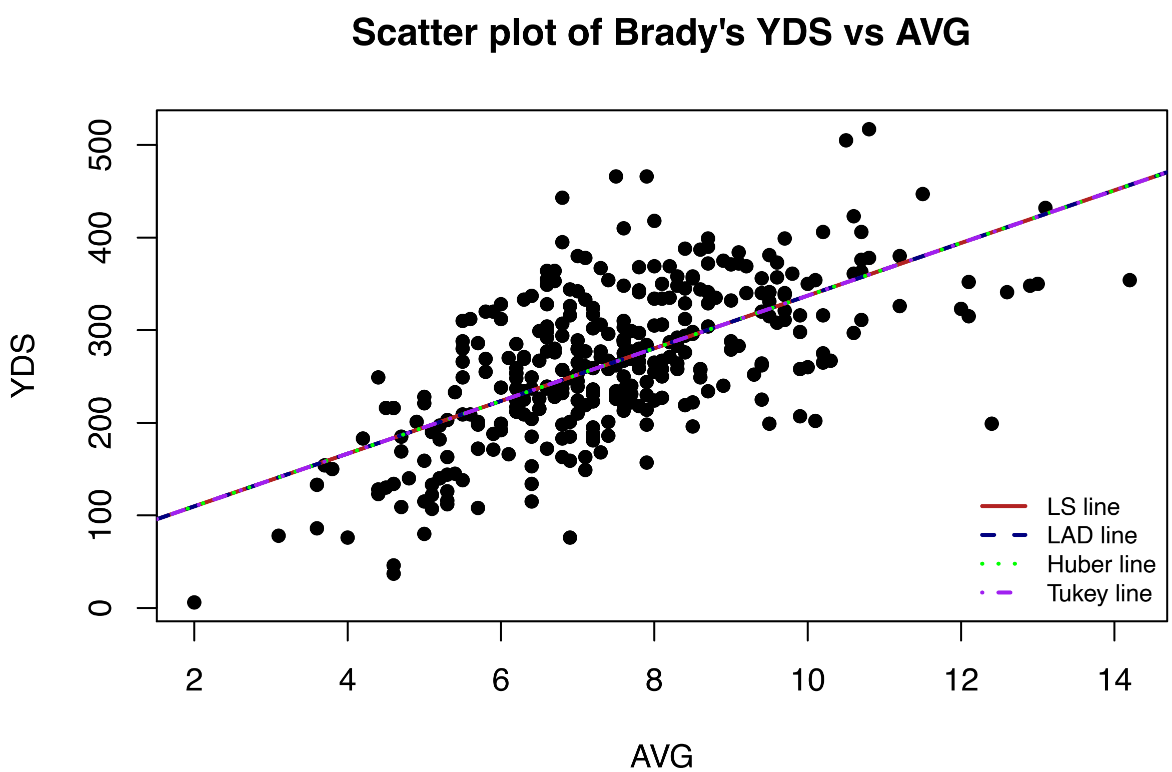
```
tukey.fn.prime <- function(resid, k=4.685) {
  val = resid - (2 * resid^3)/(k^2) + (resid^5)/(k^4)
  subr = abs(resid) > k
  val[subr] = 0
  return(val)
}

# irls function from class
IRLSresult4 <- irls(brady$YDS, brady$AVG, theta = c(264.89306, 28.43653),
                    rhoPrimeFn = tukey.fn.prime, tolerance = 1e-10,
                    maxIterations = 1000)
print(IRLSresult4)
```

```
## $theta
## [1] 264.89306  28.43653
##
## $converged
## [1] TRUE
##
## $iteration
## [1] 2
```

Based on the output, $\left(\widehat{\alpha}_{Tukey}, \widehat{\beta}_{Tukey}\right) = (264.89306, 28.43653)$

(e) Construct a scatter plot of $y =$ `YDS` versus $x =$ `AVG`. Add to this plot the four lines of best fit determined in parts (a)-(d). Be sure to distinguish between these lines with a legend and include an informative title and axis labels.

```
plot(x = brady$AVG, y= brady$YDS, main = "Scatter plot of Brady's YDS vs AVG",
    xlab = "AVG", ylab = "YDS", pch = 16)
abline(a = IRLSresult$theta[1] - IRLSresult$theta[2] * mean(brady$AVG),
       b = IRLSresult$theta[2], col = "firebrick", lwd = 2, lty = 1)
abline(a = IRLSresult2$theta[1] - IRLSresult2$theta[2] * mean(brady$AVG),
       b = IRLSresult2$theta[2], col = "navyblue", lwd = 2, lty = 2)
abline(a = IRLSresult3$theta[1] - IRLSresult3$theta[2] * mean(brady$AVG),
       b = IRLSresult3$theta[2], col = "green", lwd = 2, lty = 3)
abline(a = IRLSresult4$theta[1] - IRLSresult4$theta[2] * mean(brady$AVG),
       b = IRLSresult4$theta[2], col = "purple", lwd = 2, lty = 4)
legend("bottomright", c("LS line", "LAD line", "Huber line", "Tukey line"),
       col = c("firebrick", "navyblue", "green", "purple"),lty = 1:4, lwd = 2,
       cex = 0.75, bty = "n")
```



QUESTION 3: An Investigation of Sample Error

(a) The sub-population of interest is the set of postseason games in which Brady won. Using the `combn` function, determine all possible samples of size $n = 5$ from this population. Calculate and print out the average number of completed passing yards for the first, third, and tenth samples.

```
brady2 <- read.csv("brady.csv", header = TRUE)
bradypw <- brady2[brady2$PS == 1 & brady2$RESULT == "W", ]

pop <- bradypw$YDS
samples5 <- combn(x = pop, m = 5)
print(paste0("Mean of 1st sample: ", mean(samples5[, 1]))) # mean of 1st sample
print(paste0("Mean of 3rd sample: ",mean(samples5[, 3]))) # mean of 3rd sample
print(paste0("Mean of 10th sample: ",mean(samples5[, 10]))) # mean of 10th sample
```

```
## [1] "Mean of 1st sample: 202"

## [1] "Mean of 3rd sample: 183.4"

## [1] "Mean of 10th sample: 196.4"
```

(b) As in part (a), the sub-population of interest is the set of postseason games in which Brady won. Using the `combn` function, determine all possible samples of size $n = 30$ from this population. Calculate and print out the average number of completed passing yards for the first, third, and tenth samples.

```
samples30 <- combn(x = pop, m = 30)
print(paste0("Mean of 1st sample: ", mean(samples30[, 1]))) # mean of 1st sample
print(paste0("Mean of 3rd sample: ",mean(samples30[, 3]))) # mean of 3rd sample
print(paste0("Mean of 10th sample: ",mean(samples30[, 10]))) # mean of 10th sample
```

```
## [1] "Mean of 1st sample: 273.3"

## [1] "Mean of 3rd sample: 271.2"

## [1] "Mean of 10th sample: 272.3"
```

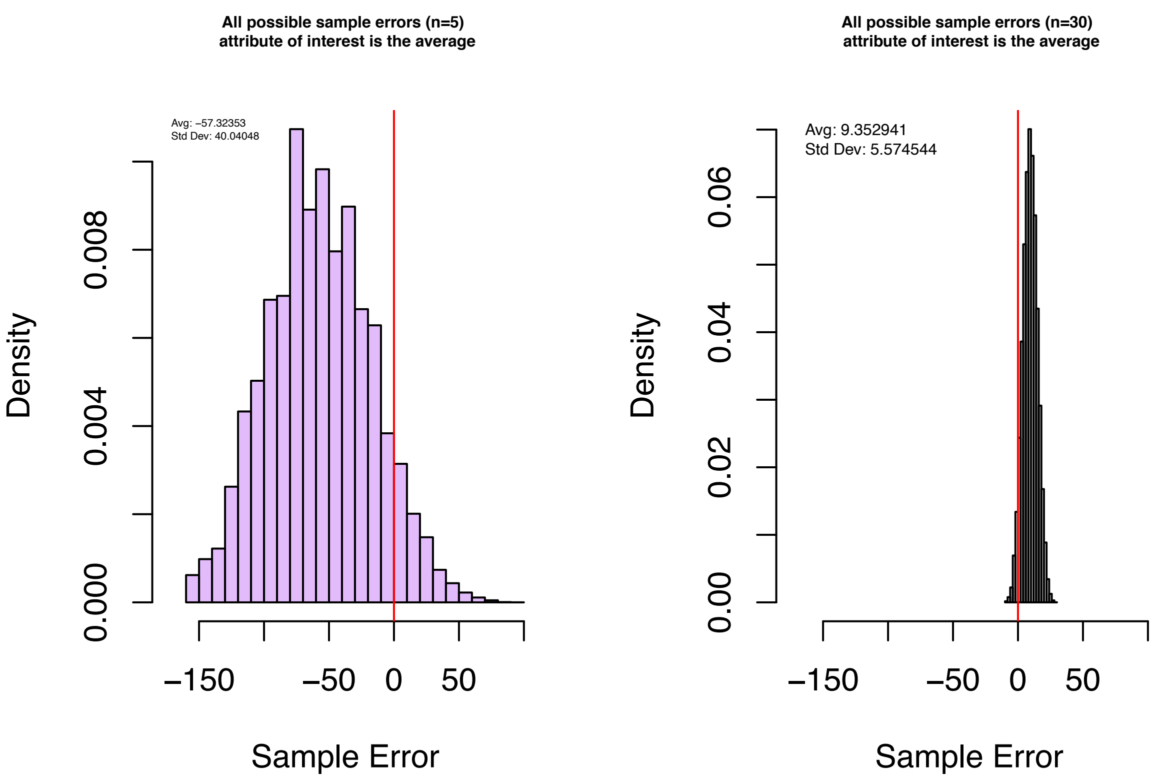
(c) In this part, the attribute of interest is the average:

$$a(\mathcal{S}) = \frac{1}{n} \sum_{u \in \mathcal{S}} y_u$$

$$a(\mathcal{P}) = \frac{1}{N} \sum_{u \in \mathcal{P}} y_u$$

```
# n = 5
avesSamp5 <- apply(samples5, MARGIN = 2, FUN = function(s) {
  mean(samples5[s])
})

# n = 30
avesSamp30 <- apply(samples30, MARGIN = 2, FUN = function(s) {
```

```
  mean(samples30[s])
})

avePop <- mean(pop)
sampleErrorsa5 <- avesSamp5 - avePop
sampleErrorsa30 <- avesSamp30 - avePop

# average and std dev of sample errors (n=5)
# avg = -57.32353, std dev = 40.04048
meanSamErra5 <- mean(sampleErrorsa5)
sdSamErra5 <- sd(sampleErrorsa5)

# average and std dev of sample errors (n=30)
# avg = 9.352941, std dev = 5.574544
meanSamErra30 <- mean(sampleErrorsa30)
sdSamErra30 <- sd(sampleErrorsa30)

par(mfrow = c(1,2))
hist(sampleErrorsa5, prob = TRUE, main = "All possible sample errors (n=5) \n attribute of interest is t
     cex.main=0.5,xlim=c(-175, 100), xlab = "Sample Error", col = adjustcolor("purple", 0.3))
abline(v = 0, col = "red")
legend("topleft", c("Avg: -57.32353", "Std Dev: 40.04048"), bty = "n", cex=0.33)
hist(sampleErrorsa30, prob = TRUE, main = "All possible sample errors (n=30) \n attribute of interest is
     cex.main=0.5, xlim=c(-175, 100), xlab = "Sample Error")
abline(v = 0, col = "red")
legend("topleft", c("Avg: 9.352941", "Std Dev: 5.574544"), bty = "n", cex=0.5)
```



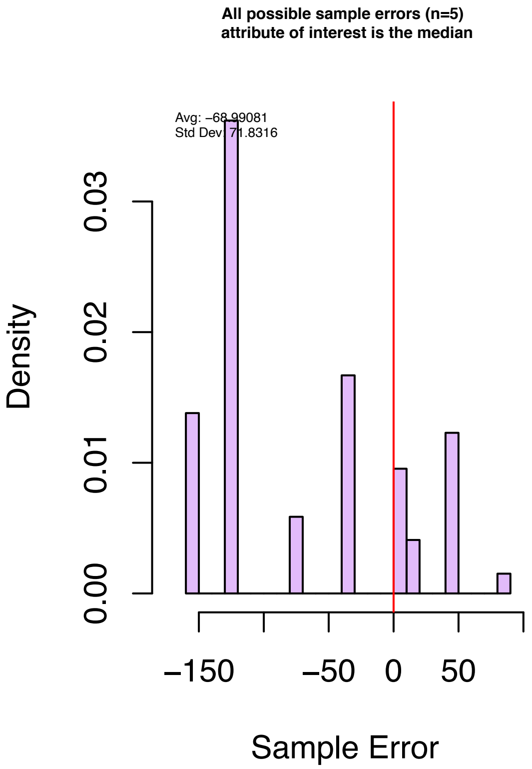
Based on the plots, sample attributes with higher sizes are concentrated to higher sample errors compared to low size samples and also suggests that lower sized sample attributes are clustered more tightly around the population since it looks that majority of them have higher absolute sample errors.

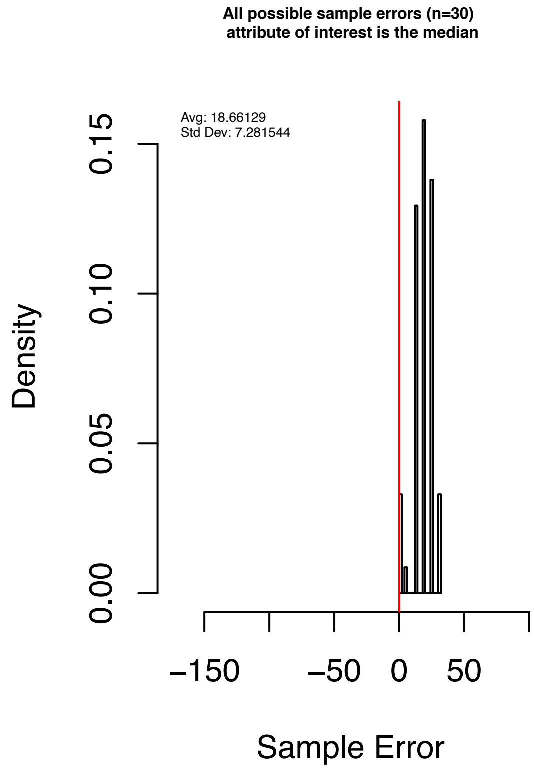
(d) In this part, the attribute of interest is the median:

$$a(\mathcal{S}) = \underset{u \in \mathcal{S}}{\text{median}}\, y_u$$

$$a(\mathcal{P}) = \underset{u \in \mathcal{P}}{\text{median}}\, y_u$$

```r
# n = 5
medSamp5 <- apply(samples5, MARGIN = 2, FUN = function(s) {
  median(samples5[s])
})

# n = 30
medSamp30 <- apply(samples30, MARGIN = 2, FUN = function(s) {
  median(samples30[s])
})

# histogram plot
medPop <- median(pop)

sampleErrorsb5 <- medSamp5 - medPop
sampleErrorsb30 <- medSamp30 - medPop

# average and std dev of sample errors (n=5)
# avg = -68.99081, std dev = 71.8316
meanSamErrb5 <- mean(sampleErrorsb5)
sdSamErrb5 <- sd(sampleErrorsb5)

# average and std dev of sample errors (n=30)
# avg = 18.66129, std dev = 7.281544
meanSamErrb30 <- mean(sampleErrorsb30)
sdSamErrb30 <- sd(sampleErrorsb30)

par(mfrow = c(1,2))
hist(sampleErrorsb5, prob = TRUE, main = "All possible sample errors (n=5) \n attribute of interest is t
     cex.main=0.5,xlim=c(-175, 100), xlab = "Sample Error", col = adjustcolor("purple", 0.3))
abline(v = 0, col = "red")
legend("topleft", c("Avg: -68.99081", "Std Dev: 71.8316"), bty = "n", cex=0.4)
hist(sampleErrorsb30, prob = TRUE, main = "All possible sample errors (n=30) \n attribute of interest is
     cex.main=0.5, xlim=c(-175, 100), xlab = "Sample Error")
abline(v = 0, col = "red")
legend("topleft", c("Avg: 18.66129", "Std Dev: 7.281544"), bty = "n", cex=0.4)
```





Based on the sample error plot with n=5, it suggests taking samples with smaller sizes have sample errors more spreaded out compared to higher sized samples. Also lower sized samples look to have a spreaded concentration compared to higher sized samples which are concentrated within [0,50]

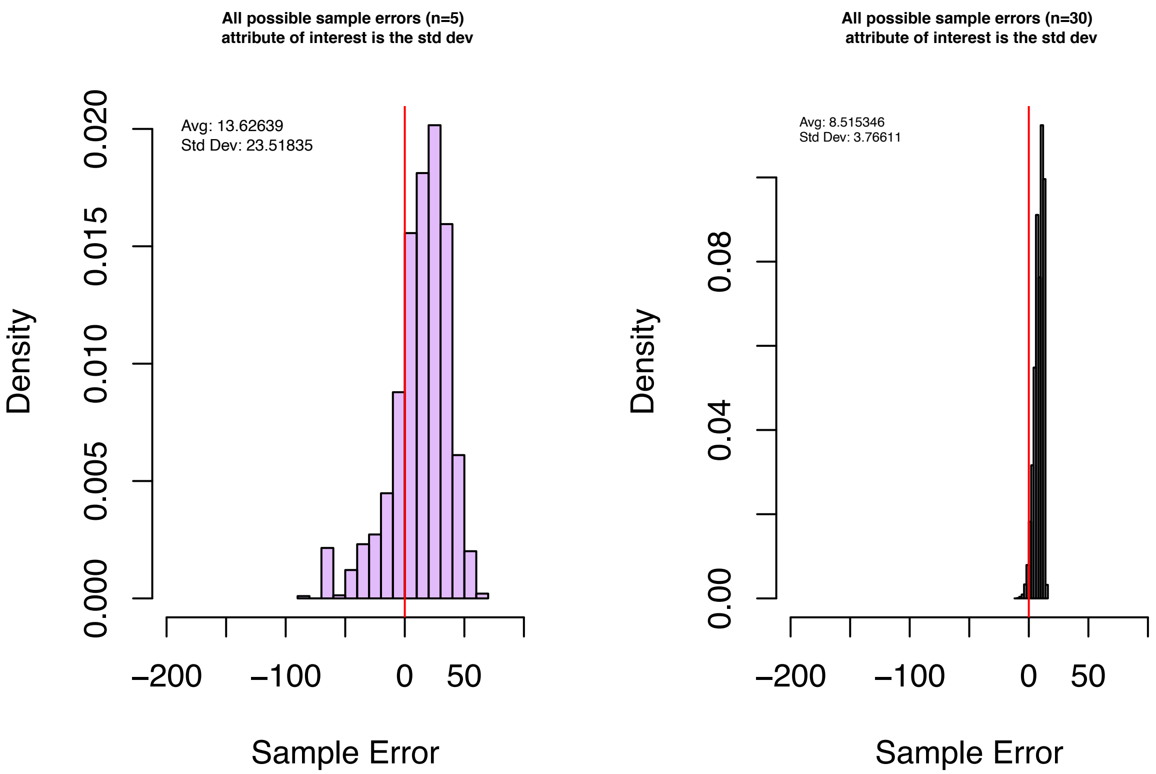
(e) In this part, the attribute of interest is the standard deviation:

$$a(\mathcal{S}) = \sqrt{\frac{1}{n}\sum_{u\in\mathcal{S}}(y_u - \overline{y})^2}$$

$$a(\mathcal{P}) = \sqrt{\frac{1}{N}\sum_{u\in\mathcal{P}}(y_u - \overline{y})^2}$$

```
# n = 5
sdSamp5 <- apply(samples5, MARGIN = 2, FUN = function(s) {
  sd(samples5[s])
})

# n = 30
sdSamp30 <- apply(samples30, MARGIN = 2, FUN = function(s) {
  sd(samples30[s])
})

# histogram plot
sdPop <- sd(pop)

sampleErrorsc5 <- sdSamp5 - sdPop
```

```
sampleErrorsc30 <- sdSamp30 - sdPop

# average and std dev of sample errors (n=5)
# avg = 13.62639, std dev = 23.51835
meanSamErrc5 <- mean(sampleErrorsc5)
sdSamErrc5 <- sd(sampleErrorsc5)

# average and std dev of sample errors (n=30)
# avg = 8.515346, std dev = 3.76611
meanSamErrc30 <- mean(sampleErrorsc30)
sdSamErrc30 <- sd(sampleErrorsc30)

par(mfrow = c(1,2))
hist(sampleErrorsc5, prob = TRUE, main = "All possible sample errors (n=5) \n attribute of interest is
     cex.main=0.5,xlim=c(-175, 100), xlab = "Sample Error", col = adjustcolor("purple", 0.3))
abline(v = 0, col = "red")
legend("topleft", c("Avg: 13.62639", "Std Dev: 23.51835"), bty = "n", cex=0.4)
hist(sampleErrorsc30, prob = TRUE, main = "All possible sample errors (n=30) \n attribute of interest i
     cex.main=0.5, xlim=c(-175, 100), xlab = "Sample Error")
abline(v = 0, col = "red")
legend("topleft", c("Avg: 8.515346", "Std Dev: 3.76611"), bty = "n", cex=0.4)
```



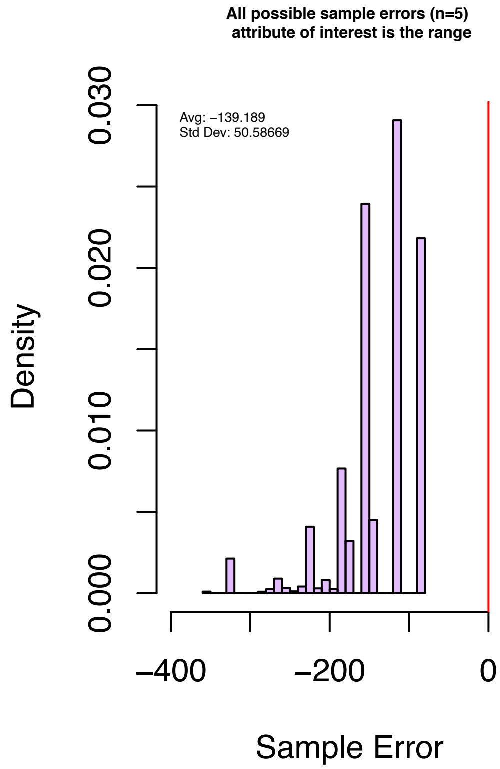
Based on the plots, it suggests that taking samples with lower sizes have sample errors more spreaded compared to samples with higher sizes. Both plots also look to be consistent as they are concentrating within [0, 50]

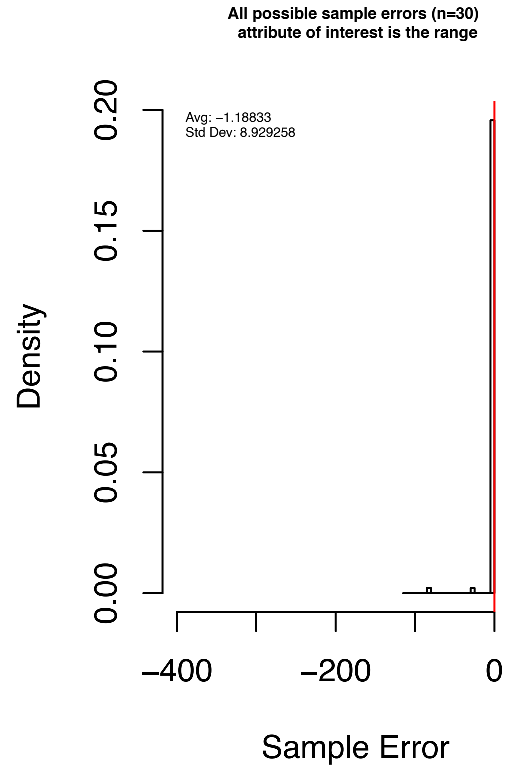
(f) In this part, the attribute of interest is the range:

$$a(\mathcal{S}) = \max_{u \in \mathcal{S}} y_u - \min_{u \in \mathcal{S}} y_u$$

$$a(\mathcal{P}) = \max_{u \in \mathcal{P}} y_u - \min_{u \in \mathcal{P}} y_u$$

```r
# n = 5
rangeSamp5 <- apply(samples5, MARGIN = 2, FUN = function(s) {
  max(samples5[s]) - min(samples5[s])
})

# n = 30
rangeSamp30 <- apply(samples30, MARGIN = 2, FUN = function(s) {
  max(samples30[s]) - min(samples30[s])
})

# histogram plot
rangePop <- max(pop) - min(pop)

sampleErrorsd5 <- rangeSamp5 - rangePop
sampleErrorsd30 <- rangeSamp30 - rangePop

# average and std dev of sample errors (n=5)
# avg = -139.189, std dev = 50.58669
meanSamErrd5 <- mean(sampleErrorsd5)
sdSamErrd5 <- sd(sampleErrorsd5)

# average and std dev of sample errors (n=30)
# avg = -1.18833, std dev = 8.929258
meanSamErrd30 <- mean(sampleErrorsd30)
sdSamErrd30 <- sd(sampleErrorsd30)

par(mfrow = c(1,2))
hist(sampleErrorsd5, prob = TRUE, main = "All possible sample errors (n=5) \n attribute of interest is
     cex.main=0.5,xlim=c(-175, 100), xlab = "Sample Error", col = adjustcolor("purple", 0.3))
abline(v = 0, col = "red")
legend("topleft", c("Avg: -139.189", "Std Dev: 50.58669"), bty = "n", cex=0.4)
hist(sampleErrorsd30, prob = TRUE, main = "All possible sample errors (n=30) \n attribute of interest i
     cex.main=0.5, xlim=c(-175, 100), xlab = "Sample Error")
abline(v = 0, col = "red")
legend("topleft", c("Avg: -1.18833", "Std Dev: 8.929258"), bty = "n", cex=0.4)
```





Based on the plots, it looks that there is no consistency between the plots, however it looks that the samples are concentrated to the right. Also the sample attributes of lower size look to be more clustered around the population since their absolute sample errors are higher.