Problem Set 11

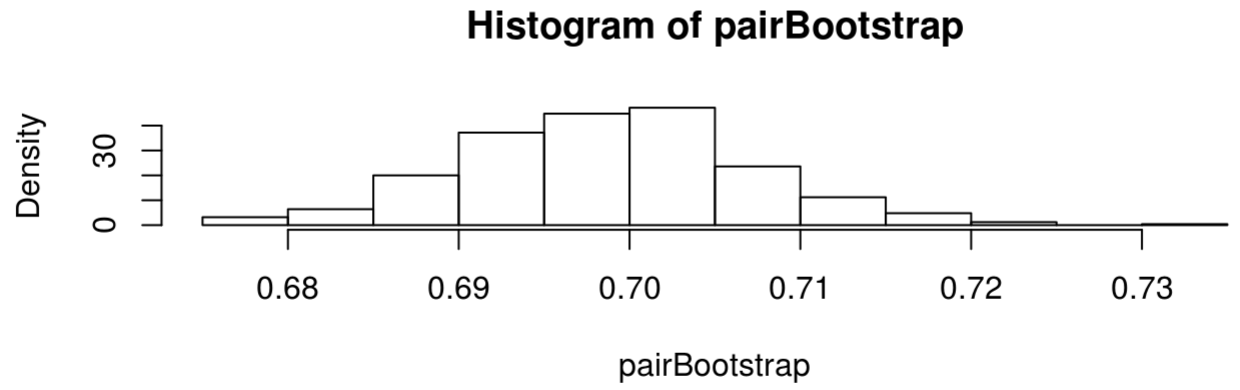
Ian McGroarty

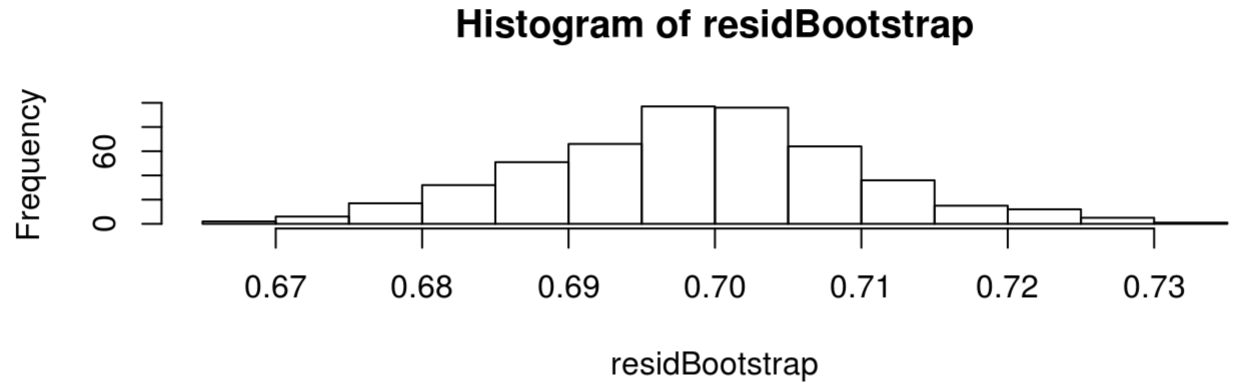
19APRIL2020

Problem 1:

We can see from the histograms below that the pair bootstrap has a longer tail. In gerneral it sees to be a bit more skewed. Though both are centerted around the same mean.

```
#### Set up data ####
R = c(68, 77, 299, 220, 142, 287, 276, 115, 64, 206, 222, 205, 233, 228, 188, 132, 285, 188, 224
, 121, 311, 166, 248, 161, 226, 67, 201, 267, 121, 301, 244, 222, 195, 203, 210, 275, 286, 275,
304, 214)

S = c(56, 62, 445, 279, 138, 428, 319, 102, 51, 289, 351, 282, 310, 266, 256, 144, 447, 186, 389
, 113, 412, 176, 313, 162, 368, 54, 214, 429, 115, 407, 265, 301, 234, 229, 270, 478, 419, 490,
430, 235)

salmon <- as.data.frame(R)
salmon$S <- S
salmon$R2 <- 1/R
salmon$S2 <- 1/S

#### First regression #####
origRgrsn <- lm(formula = R2 ~ S2, data=salmon)

#### Bootstrap Bill Turner ####
  # Set Parameters
    B <- 500
    pairBootstrap = rep(0,B)
      residBootstrap = rep(0,B)

  for (j in 1:B) {
    #   bootstrap covariates and response
      ## Get Sample Data - sample by row number
          resampledObs <- base::sample(seq(1:length(salmon[,1])),length(salmon[,1]), replace = T
RUE)
          newDataSet = salmon[resampledObs,]
        ## Run regression and store coefficient
            newRgrsn = lm(formula = R2 ~ S2, data=newDataSet)
            pairBootstrap[j] = newRgrsn$coefficients[2]

    #   bootstrap residuals
          ## The the residual for each observation.
            resampledResids <- origRgrsn$residuals[resampledObs]
          ## Get prediction
            salmon$alteredRspns = resampledResids + predict(origRgrsn)
        newResidRgrsn = lm(alteredRspns ~ S2,data=salmon)
        residBootstrap[j] = newResidRgrsn$coefficients[2]

  }


#plot(S,R)
par(mfrow = c(2,1))
hist(pairBootstrap, freq=FALSE)
hist(residBootstrap)
```





Problem 2

Generate 100 samples $X_1, \cdots, X_{20}$ from a normal population $N(\theta, 1)$ with $\theta = 1$.

(a) For each sample compute the bootstrap and jackknife estimate for variance for $\hat{\theta} = \bar{X}$ and compute the mean and standard deviation of these variance estimates over the 100

samples.

```r
B <- 1000
theta <- c()

##  Build the functions
  bootstrap.var <- function(DATA,B){
      for (i in 1:B){
          # Make sure the data is in vector form
            vctrzdData <- as.vector(DATA)
          # Sample of size n with replacement from data
            btstrpSmpl = base::sample(vctrzdData,length(vctrzdData), replace = TRUE)
          # Sort
            btstrpSmpl.ordered <- sort(btstrpSmpl)
          # Take the mean of the middle 4 numbers
          # theta[i] <- mean(btstrpSmpl.ordered[3:6])
            theta[i] <- mean(btstrpSmpl.ordered)
        }
    # Take the mean of the B thetas
      theta.mean <- mean(theta)
    # Take the variance of the B thetas (var(theta) also works)
      theta.var <- (1/(B-1))*(sum((theta-theta.mean)^2))
    # Return
      return(c(theta.mean,theta.var))
  }

  jackknife.var <- function(DATA){
    # Clear theta
      theta <- c()
    # Get the length of DATA
      n <- length(DATA)
    # Get the theta_j values for the jackknife samples.
      for (j in 1:n){
        # Generate the sample excluding the ith value of X1
          btstrpSmpl <- DATA[-j]
        # Take the mean of the sample.
          theta[j] <- mean(btstrpSmpl)
      }
    # Compute theta(.)
      theta.dot <- mean(theta)
    # Jackknife variance
      dif <- (theta - theta.dot)^2
      var.jackknife <- sqrt(((n-1)/n)*sum(dif))
    # Return
        return(c(theta.dot,var.jackknife))
    # print(paste0("Jackknife: Mean = ",theta.dot," Variance = ", var.jackknife))
  }

## Run
jack.mean <- c()
jack.var <- c()
boot.mean <- c()
boot.var <- c()
for ( i in 1:100){
  # Generate X1
```

```
    X1 <- rnorm(n=20,mean=1,sd=1)
  # Bootstrap
    boot <- bootstrap.var(X1,B)
      boot.mean[i] <- boot[1]
      boot.var[i] <- boot[2]
  # Jackknife
  jack <- jackknife.var(X1)
    jack.mean[i] <- jack[1]
    jack.var[i] <- jack[2]
}

print(paste0("Bootstrap: B = ",B, " Mean = ",mean(boot.mean), " mean(Variance) = ", mean(boot.va
r), " St. Dev.(variance) = ",sd(boot.var)))
```

```
## [1] "Bootstrap: B = 1000 Mean = 0.994970100498525 mean(Variance) = 0.047877760262376 St. Dev.
(variance) = 0.0170191611810896"
```

```
print(paste0("Jackknife: Mean = ",mean(jack.mean)," mean(Variance) = ", mean(jack.var)," St. De
v.(variance) = ",sd(jack.var)))
```

```
## [1] "Jackknife: Mean = 0.994536074257312 mean(Variance) = 0.222451568199411 St. Dev.(varianc
e) = 0.0409148554100715"
```

(b) Repeat (a) for $\hat{\theta} = \bar{X}^2$

```r
B <- 1000
theta <- c()

bootstrap.var <- function(DATA,B){
    for (i in 1:B){
        # Make sure the data is in vector form
          vctrzdData <- as.vector(DATA)
        # Sample of size n with replacement from data
          btstrpSmpl = base::sample(vctrzdData,length(vctrzdData), replace = TRUE)
        # Sort
          btstrpSmpl.ordered <- sort(btstrpSmpl)
        # Take the mean of the middle 4 numbers
        # theta[i] <- mean(btstrpSmpl.ordered[3:6])
          theta[i] <- mean(btstrpSmpl.ordered)
      }
  # Take the mean of the B thetas
    theta.mean <- (mean(theta))^2
  # Take the variance of the B thetas (var(theta) also works)
    theta.var <- (1/(B-1))*(sum((theta-theta.mean)^2))
  # Return
    return(c(theta.mean,theta.var))
}

jackknife.var <- function(DATA){
  # Clear theta
    theta <- c()
  # Get the length of DATA
    n <- length(DATA)
  # Get the theta_j values for the jackknife samples.
    for (j in 1:n){
      # Generate the sample excluding the ith value of X1
        btstrpSmpl <- DATA[-j]
      # Take the mean of the sample.
        theta[j] <- (mean(btstrpSmpl))^2
    }
  # Compute theta(.)
    theta.dot <- mean(theta)
  # Jackknife variance
    dif <- (theta - theta.dot)^2
    var.jackknife <- sqrt(((n-1)/n)*sum(dif))
  # Return
      return(c(theta.dot,var.jackknife))
  # print(paste0("Jackknife: Mean = ",theta.dot," Variance = ", var.jackknife))
}

jack.mean <- c()
jack.var <- c()
boot.mean <- c()
boot.var <- c()
for ( i in 1:100){
  # Generate X1
    X1 <- rnorm(n=20,mean=1,sd=1)
  # Bootstrap
```

```
    boot <- bootstrap.var(X1,B)
      boot.mean[i] <- boot[1]
      boot.var[i] <- boot[2]
  # Jackknife
  jack <- jackknife.var(X1)
    jack.mean[i] <- jack[1]
    jack.var[i] <- jack[2]
}

print(paste0("Bootstrap: B = ",B, " Mean = ",mean(boot.mean), " mean(Variance) = ", mean(boot.va
r), " St. Dev.(variance) = ",sd(boot.var)))
```

```
## [1] "Bootstrap: B = 1000 Mean = 1.1038158182034 mean(Variance) = 0.110581802674557 St. Dev.(v
ariance) = 0.126440650553676"
```

```
print(paste0("Jackknife: Mean = ",mean(jack.mean)," mean(Variance) = ", mean(jack.var)," St. De
v.(variance) = ",sd(jack.var)))
```

```
## [1] "Jackknife: Mean = 1.10764142696516 mean(Variance) = 0.449888186545517 St. Dev.(variance)
= 0.114055888362321"
```

Problem 3

Find the number of bootstraps necessary to accurately calculate a 95% confidence interval in the difference of two population means. That is, simulate n values of X, where $X_i N(_1, _1^2) and m values of Y , where $Y_j N(_2, _2^2) and calculate a 95% confidence interval in $\mu_1 - \mu_2$. Now calculate a 95% confidence interval for $\mu_1 - \mu_2$ using the bootstrap techniques learned in this module. How large does B have to be until you accurately and dependably reproduce the confidence interval you originally calculated?

```
# Set parameters
  B <- 200
  n <- 200
  mu1 <- 1
  sig1 <- 1
  m <- 20
  mu2 <- 1
  sig2 <- 2

# Generate the true samples
  X <- rnorm(n,mean=mu1,sd=sig1)
  Y <- rnorm(m,mean=mu2,sd=sig2)

# Theoretical Confidence interval
  # Sample variance (Larsen & Marx pg 475)
    sample.var <- function(DATA){
      (length(DATA)*(sum(DATA^2)) - (sum(DATA))^2)/(length(DATA)*(length(DATA)-1))
    }
  # T statistic
    t <- qt(0.025,df=(n+m-2), lower.tail = FALSE)
    term <- t*(sqrt((sample.var(X)/n)+(sample.var(Y)/m)))
  # Confidence interval
    print(paste0("The 95% Confidence interval for X-Y is (",
                 mean(Y)-mean(Y)-term,",", mean(Y)-mean(Y)+term,")"))
```

```
## [1] "The 95% Confidence interval for X-Y is (-0.665609515885879,0.665609515885879)"
```

```
# Bootstrap
    mean.X <- c()
    mean.Y <- c()
  for (i in 1:B){
      # Sample of size n with replacement from data
        btstrpSmpl.X = base::sample(X,length(X), replace = TRUE)
        btstrpSmpl.Y = base::sample(Y,length(Y), replace = TRUE)
      # Mean
        mean.X[i] <- mean(btstrpSmpl.X)
        mean.Y[i] <- mean(btstrpSmpl.Y)
    }
# Calculate theta
  theta <- mean.X-mean.Y
# Order Theta
  theta.ordered <- sort(theta)
# Where do the intervales lie
  min <- (B-(0.95*B))/2
# Interval
  print(paste0("The 95% Confidence interval for X-Y is (",
                theta.ordered[min],",", theta.ordered[(B-min)],")"))
```

```
## [1] "The 95% Confidence interval for X-Y is (-1.54235337620315,-0.116471898835519)"
```

```
# Set parameters
  B <- 200
  n <- 200
  mu1 <- 1
  sig1 <- 1
  m <- 20
  mu2 <- 1
  sig2 <- 2

# Generate the true samples
  X <- rnorm(n,mean=mu1,sd=sig1)
  Y <- rnorm(m,mean=mu2,sd=sig2)

# Theoretical Confidence interval
  # Sample variance (Larsen & Marx pg 475)
    sample.var <- function(DATA){
      (length(DATA)*(sum(DATA^2)) - (sum(DATA))^2)/(length(DATA)*(length(DATA)-1))
    }
  # T statistic
    t <- qt(0.025,df=(n+m-2), lower.tail = FALSE)
    term <- t*(sqrt((sample.var(X)/n)+(sample.var(Y)/m)))
  # Confidence interval
    print(paste0("The 95% Confidence interval for X-Y is (",
                 mean(Y)-mean(Y)-term,",", mean(Y)-mean(Y)+term,")"))
```

```
## [1] "The 95% Confidence interval for X-Y is (-0.955542866058812,0.955542866058812)"
```

```
# Bootstrap
    bootstrap.twosamp.confint <- function(DATA1,DATA2,B){
        mean.X <- c()
        mean.Y <- c()
        theta <- c()
        min <- c()
        max <- c()
      for (i in 1:B){
          # Sample of size n with replacement from data
            btstrpSmpl.X = base::sample(DATA1,length(DATA1), replace = TRUE)
            btstrpSmpl.Y = base::sample(DATA2,length(DATA2), replace = TRUE)
          # Mean
            mean.X[i] <- mean(btstrpSmpl.X)
            mean.Y[i] <- mean(btstrpSmpl.Y)
          # Calculate theta
            theta[i] <- mean.X[i]-mean.Y[i]
        }

    # Order Theta
      theta.ordered <- sort(theta)
    # Where do the intervals lie
      int <- (B-(0.95*B))/2
    # Interval
      min <- theta.ordered[int]
      max <- theta.ordered[(B-int)]
      return(c(min,max))
    }

# Lets do this for B
  bootstrap.conf <- bootstrap.twosamp.confint(X,Y,B)
    print(paste0("The 95% Confidence interval for X-Y is (",
                bootstrap.conf[1],",", bootstrap.conf[2],")"))
```

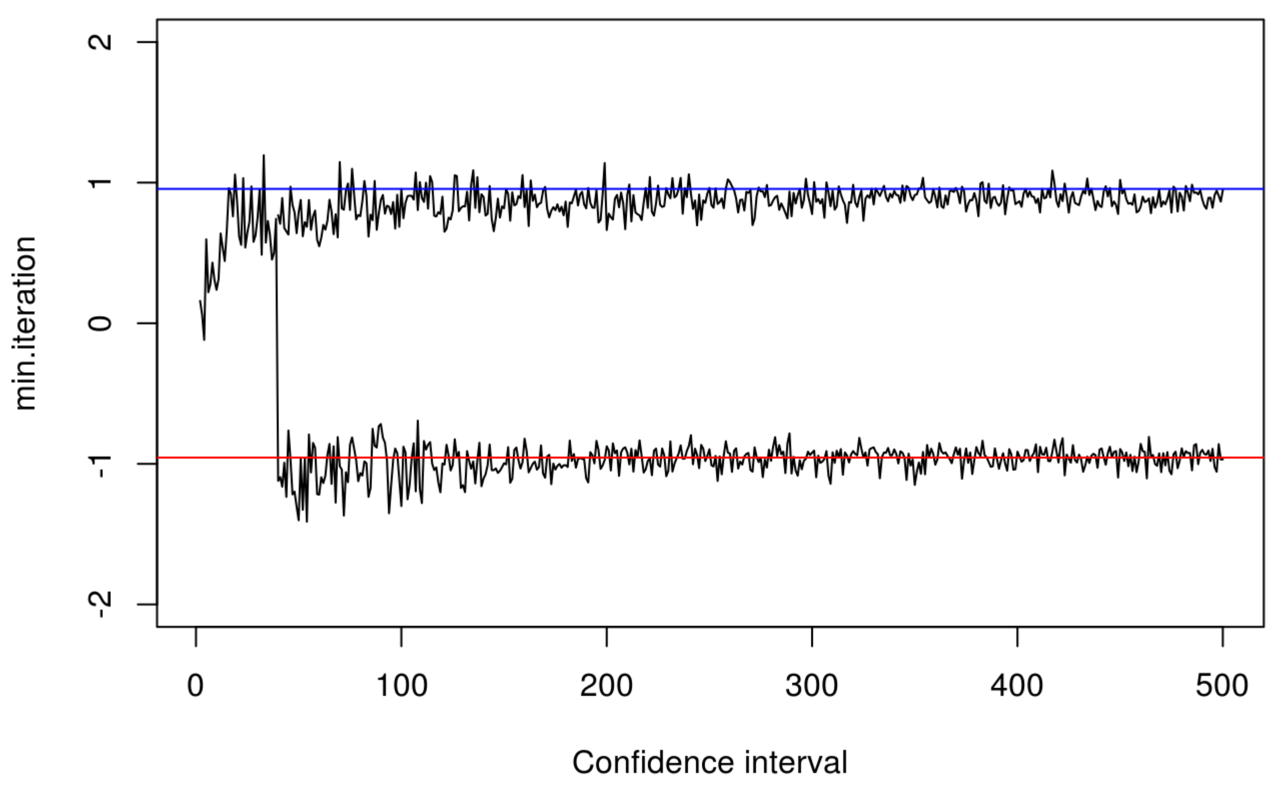
```
## [1] "The 95% Confidence interval for X-Y is (-0.84968537443988,0.863594086427095)"
```

```
# For the plot
  min.iteration <- c()
  max.iteration <- c()
  for (j in 1:500){
   iteration <- bootstrap.twosamp.confint(X,Y,j)
   min.iteration[j] <- iteration[1]
   max.iteration[j] <- iteration[2]
  }


  plot(min.iteration, type='l',
       ylim=c(-2,2),xlab="Confidence interval")
  lines(max.iteration, type='l')
  abline(h=(mean(Y)-mean(Y)-term), type='l', col='red')
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): graphical
## parameter "type" is obsolete
```

```
  abline(h=(mean(Y)-mean(Y)+term), type='l', col='blue')
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): graphical
## parameter "type" is obsolete
```



From the plot above, it looks like the confidence interval is pretty stable once it reaches (approximately) B= 40.