Inference_Hw3

Warren Geither

10/6/2020

Problem 1

$$Y \sim N(\mu, \sigma^2)$$
$$X = \frac{Y-\mu}{\sigma} = g(y)$$
$$\implies g^{-1}(x) = x\sigma + \mu$$
$$\implies \frac{d}{dx}(x\sigma + \mu)$$
$$= \sigma$$

Using Theorem 2.1.5,

$$f_X(x) = f_Y(g^{-1}(x))|\frac{d}{dx}(g^{-1}(x))|$$
$$= \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-((x\sigma+\mu)-\mu)^2)}{2\sigma^2}} |\sigma|$$
$$= \frac{1}{\sqrt{2\pi}} e^{\frac{-(x\sigma)^2}{2\sigma^2}}$$
$$= \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \sim N(0,1) \blacksquare$$

Problem 2

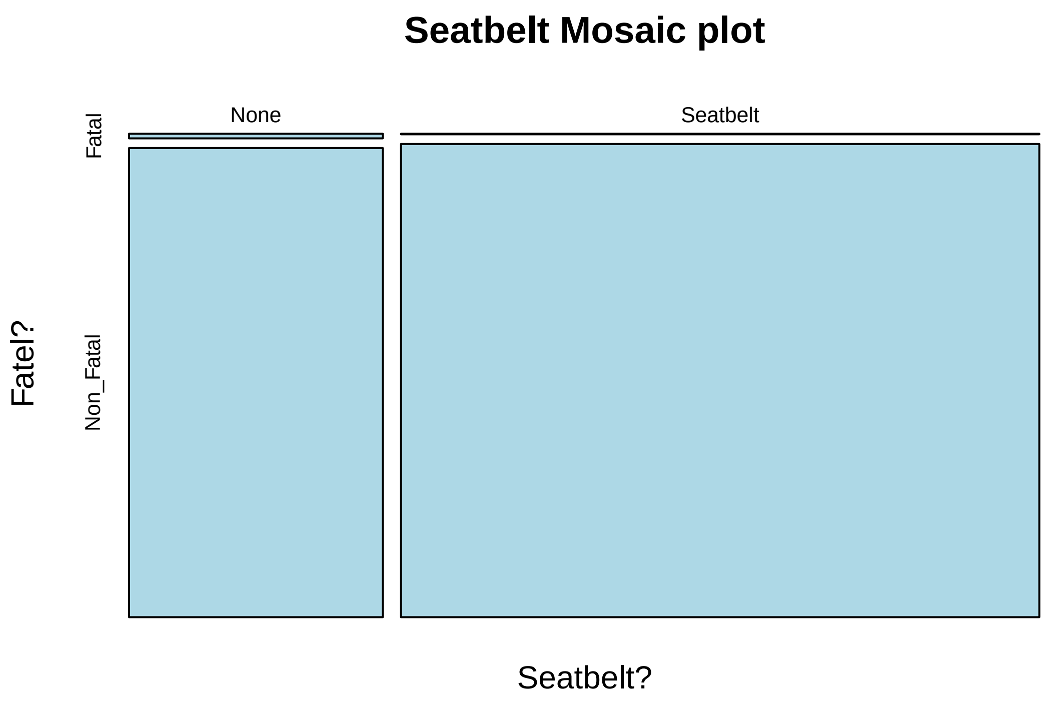
```
# create df
seatbelt_df <- data.frame(Fatal = c(1601,510)
                          , Non_Fatal = c(162527,412368))

# add rownames
rownames(seatbelt_df) <- c("None", "Seatbelt")

# print table
knitr::kable(seatbelt_df)
```

	Fatal	Non_Fatal
None	1601	162527
Seatbelt	510	412368

```r
# print mosiac plot
mosaicplot(seatbelt_df
           , main = 'Seatbelt Mosaic plot'
           , xlab='Seatbelt?'
           , ylab='Fatel?'
           , color='light blue')
```



The mosiac plot shows that the number of people who wore a seatbelt had slightly higher non fatal accidents and fewer fatal accidents than people who did not where a seatbelt.

```r
# calculate proportions
fatal_given_seatbelt <- seatbelt_df[2,1]/ sum(seatbelt_df[2,1] + seatbelt_df[2,2])

fatal_given_none <- seatbelt_df[1,1]/ sum(seatbelt_df[1,1] + seatbelt_df[1,2])

# absolute difference in prop
diff_in_prop <- fatal_given_seatbelt - fatal_given_none

# relative risk
rel_risk <- fatal_given_none/fatal_given_seatbelt

# calculate odds
odds_fatal_seatbelt <- seatbelt_df[2,1]/seatbelt_df[2,2]
odds_fatal_none <- seatbelt_df[1,1]/seatbelt_df[1,2]

# calculate odds ratio
```

```
odds_ratio <- odds_fatal_none/odds_fatal_seatbelt

# proportions
print(paste0("Proportion fatal given seatbelt: ", round(fatal_given_seatbelt,4)))
```

```
## [1] "Proportion fatal given seatbelt: 0.0012"
```

```
print(paste0("Proportion fatal given none: ", round(fatal_given_none,4)))
```

```
## [1] "Proportion fatal given none: 0.0098"
```

```
# fix odds ratio
print(paste0("Difference in Proportion: ", round(diff_in_prop,4)))
```

```
## [1] "Difference in Proportion: -0.0085"
```

```
print(paste0("Relative Risk: ", round(rel_risk,4)))
```

```
## [1] "Relative Risk: 7.897"
```

```
print(paste0("Odds Ratio: ", odds_ratio))
```

```
## [1] "Odds Ratio: 7.96490487191449"
```

- The difference in proportion tells us that proportion of people who wore a seatbelt and were in a fatal accident is 0.0085 lower than the proportion of people who were in a fatal accident and did not wear a seatbelt
- The relative risk tells us that the proportion of people who were in a fatal accident given they did got where a seatbelt is 7.96 times higher than the proportion of people who were in a fatal accident who also wore a seatbelt
- The odds ratio tells us that the odds of being in a fatal accident while not wearing a seatbelt are 7.96 times higher than the odds of being in a fatal accident while wearing a seatbelt

Problem 3

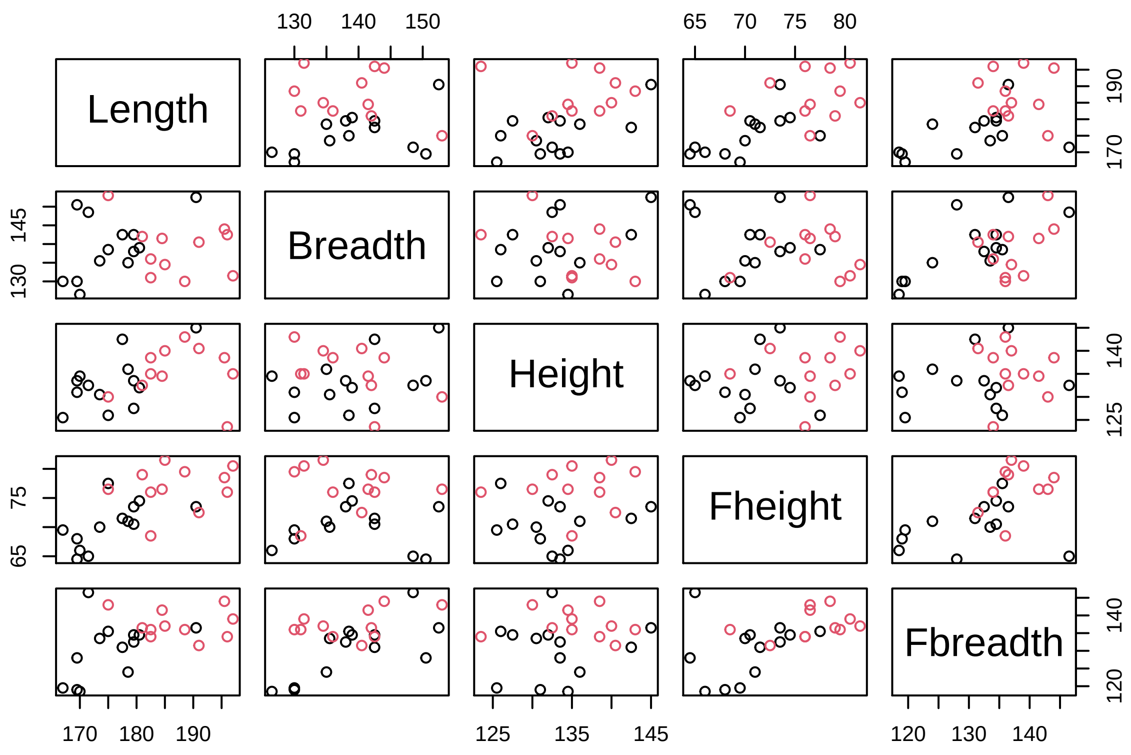
3.1)

```
# read data in
skull_df <- read.csv("TSkull_19.csv")

# split training and test set
training_df <- skull_df %>% filter(Holdout == 0) %>% select(-Holdout)
testing_df <- skull_df %>% filter(Holdout == 1) %>% select(-Holdout)
```

3.2)

```r
# create coorlation matrix
pairs(training_df[,1:5], col=training_df$Type)
```



Classification rank: 1.) Length 2.) Fheight 3.) Height 4.) Fbreadth 5.) Breadth

3.3)

Calculating the difference in means for an appropriate effect size.

```r
# calculate mean difference for unstandardized effect size
mean_summary_df <- training_df %>%
                    group_by(Type) %>%
                    summarize(length = mean(Length)
                              , breadth = mean(Breadth)
                              , height = mean(Height)
                              , fheight = mean(Fheight)
                              , fbreadth = mean(Fbreadth))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# tidy version of mean summary
mean_df <- pivot_longer(mean_summary_df, cols = 2:6, names_to = "Mean")

# order the data
```

```
ordered_df <- mean_df[order(mean_df$Mean),]

# calculate diff in means
diff_mean_df <- ordered_df %>%
    group_by(Mean) %>%
    mutate(Mean_Diff = value - lag(value))

# print table
knitr::kable(diff_mean_df)
```

Type	Mean	value	Mean_Diff
1	breadth	139.15385	NA
2	breadth	138.77273	-0.3811189
1	fbreadth	130.26923	NA
2	fbreadth	137.50000	7.2307692
1	fheight	70.38462	NA
2	fheight	76.81818	6.4335664
1	height	133.07692	NA
2	height	135.54545	2.4685315
1	length	175.53846	NA
2	length	187.13636	11.5979021

3.4)

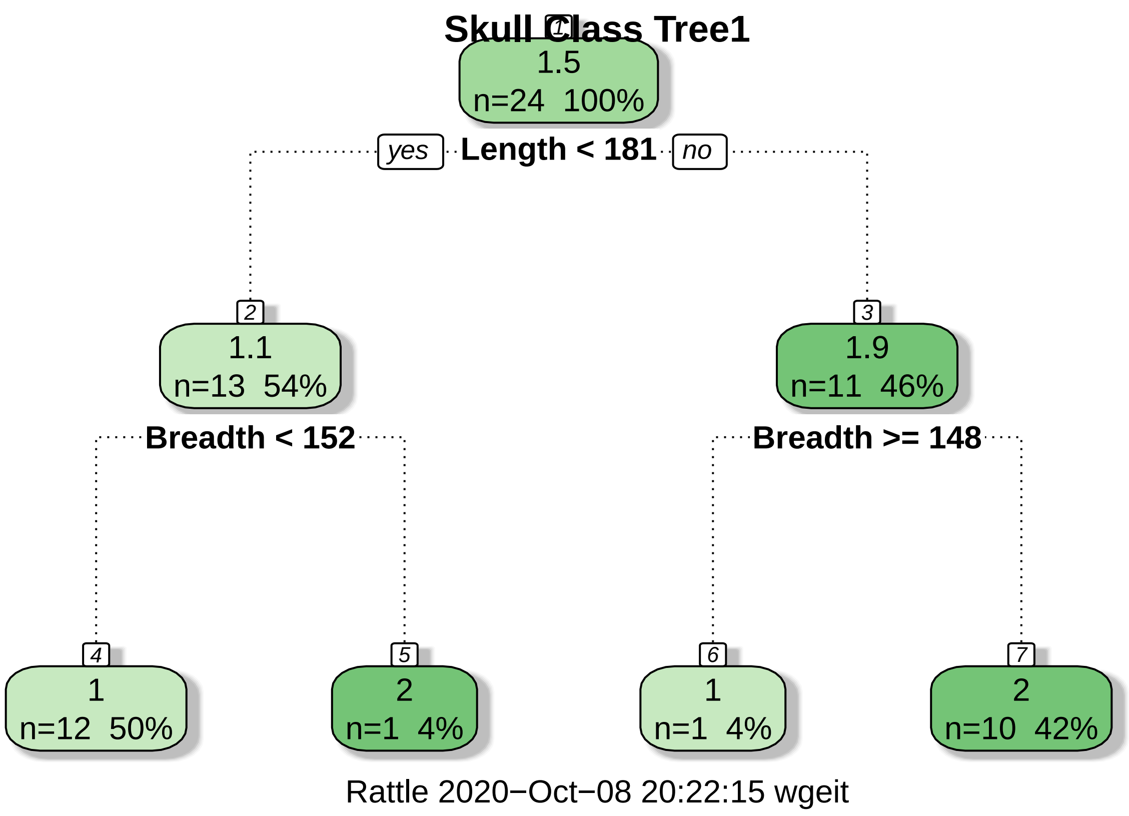
```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
# tree 1
skull_tree1 <- rpart(Type~.
                     , data = training_df
                     , control = rpart.control(minsplit = 2)
                     )

fancyRpartPlot(skull_tree1, main = "Skull Class Tree1")
```

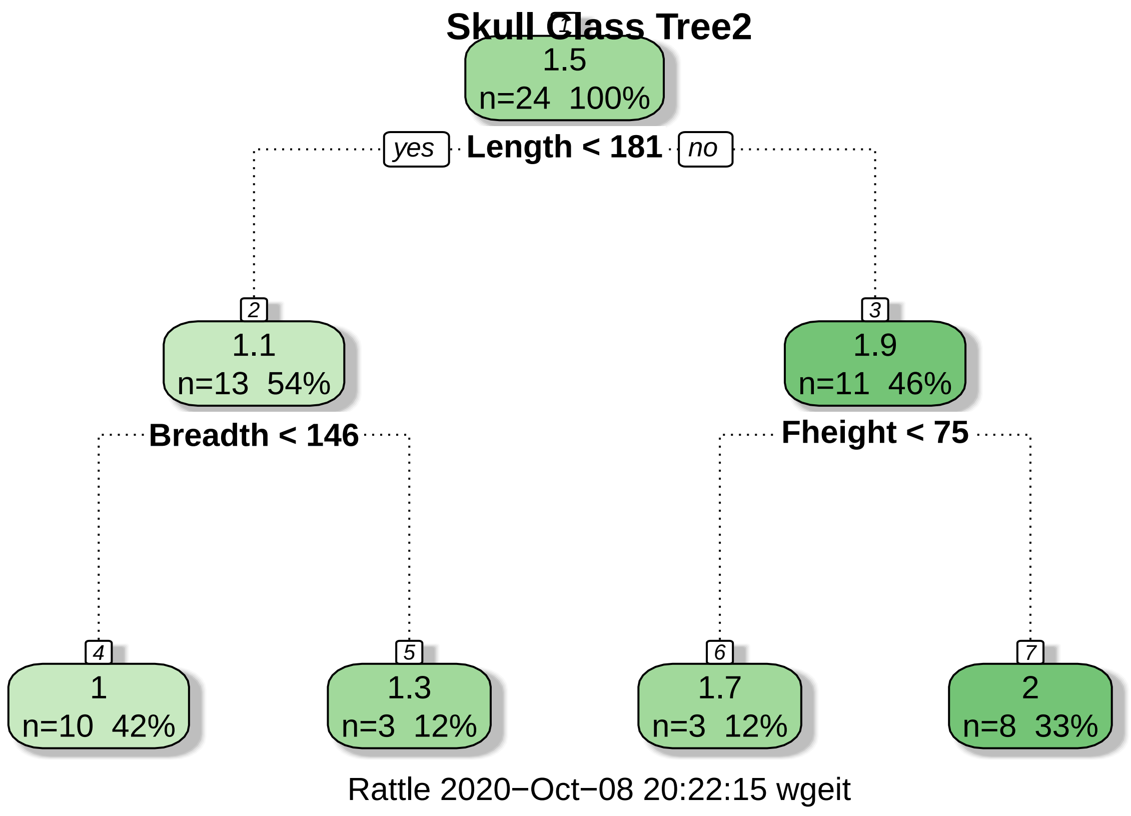


```r
# Root node error: 5.9583/24 = 0.24826
printcp(skull_tree1)
```

```
##
## Regression tree:
## rpart(formula = Type ~ ., data = training_df, control = rpart.control(minsplit = 2))
##
## Variables actually used in tree construction:
## [1] Breadth Length
##
## Root node error: 5.9583/24 = 0.24826
##
## n= 24
##
##         CP nsplit rel error  xerror     xstd
## 1 0.69250      0   1.00000 1.06406 0.042076
## 2 0.15492      1   0.30750 0.81380 0.309992
## 3 0.15257      2   0.15257 0.83916 0.333912
## 4 0.01000      3   0.00000 0.83916 0.333912
```

```r
# tree2
skull_tree2 <- rpart(Type~.
                     , data = training_df
                     , control = rpart.control(minsplit = 10)
                     )
```

```
fancyRpartPlot(skull_tree2, main = "Skull Class Tree2")
```

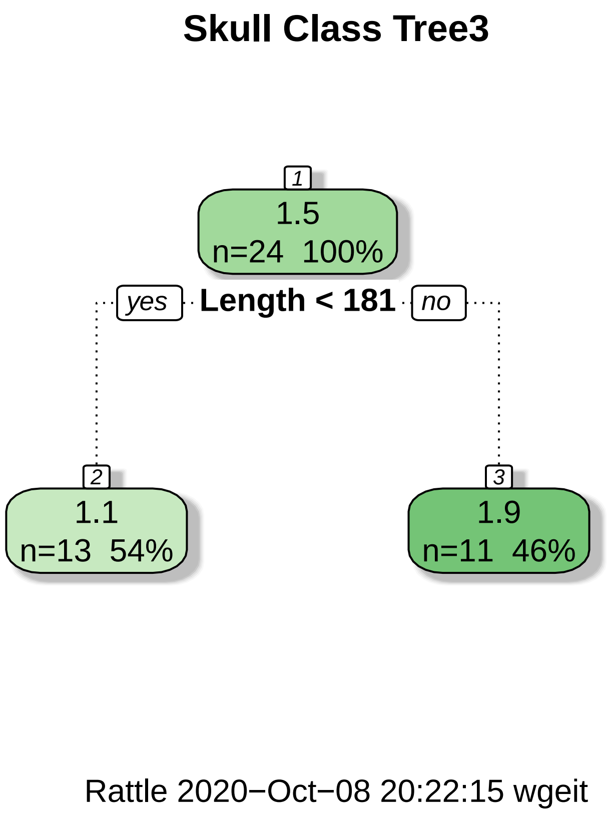


```
# 5.9583/24 = 0.24826
printcp(skull_tree2)
```

```
##
## Regression tree:
## rpart(formula = Type ~ ., data = training_df, control = rpart.control(minsplit = 10))
##
## Variables actually used in tree construction:
## [1] Breadth Fheight Length
##
## Root node error: 5.9583/24 = 0.24826
##
## n= 24
##
##         CP nsplit rel error  xerror     xstd
## 1 0.692503      0   1.00000 1.01458 0.035955
## 2 0.043034      1   0.30750 0.82901 0.320231
## 3 0.040687      2   0.26446 0.91375 0.327780
## 4 0.010000      3   0.22378 0.91375 0.327780
```

```r
# tree 3
skull_tree3 <- rpart(Type~.
                     , data = training_df
                     , control = rpart.control(minsplit = 20)
                     )

fancyRpartPlot(skull_tree3, main = "Skull Class Tree3")
```

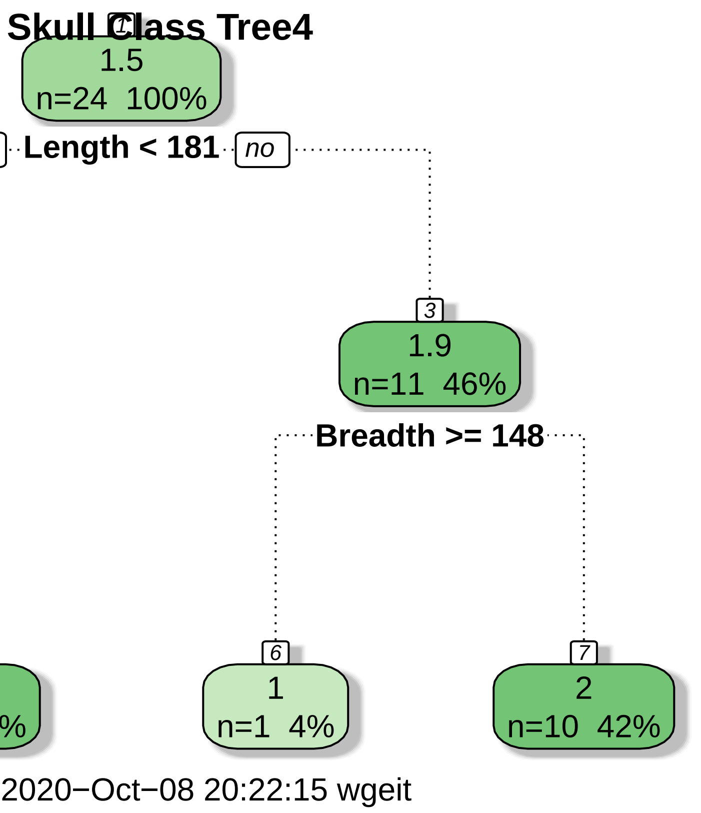


```r
# 5.9583/24 = 0.24826
printcp(skull_tree3)
```

```
##
## Regression tree:
## rpart(formula = Type ~ ., data = training_df, control = rpart.control(minsplit = 20))
##
## Variables actually used in tree construction:
## [1] Length
##
## Root node error: 5.9583/24 = 0.24826
##
## n= 24
##
##       CP nsplit rel error  xerror     xstd
## 1 0.6925      0    1.0000 1.13345 0.049588
## 2 0.0100      1    0.3075 0.68086 0.292760
```

```r
# tree 4
skull_tree4 <- rpart(Type~.
                     , data = training_df
                     , control = rpart.control(minsplit = 1
                                              , minbucket = 1
                                              , maxdepth = 30
                                              , cp = 0)
                     )

fancyRpartPlot(skull_tree4, main = "Skull Class Tree4")
```



```r
# Root node error: 5.9583/24 = 0.24826
printcp(skull_tree4)
```

```
##
## Regression tree:
## rpart(formula = Type ~ ., data = training_df, control = rpart.control(minsplit = 1,
##     minbucket = 1, maxdepth = 30, cp = 0))
##
## Variables actually used in tree construction:
## [1] Breadth Length
##
## Root node error: 5.9583/24 = 0.24826
##
## n= 24
```

```
##
##         CP nsplit rel error xerror     xstd
## 1 0.69250      0   1.00000 1.1603 0.040461
## 2 0.15492      1   0.30750 1.1493 0.353566
## 3 0.15257      2   0.15257 1.1748 0.373717
## 4 0.00000      3   0.00000 1.1748 0.373717
```

All the trees have the same root node error of 0.24826

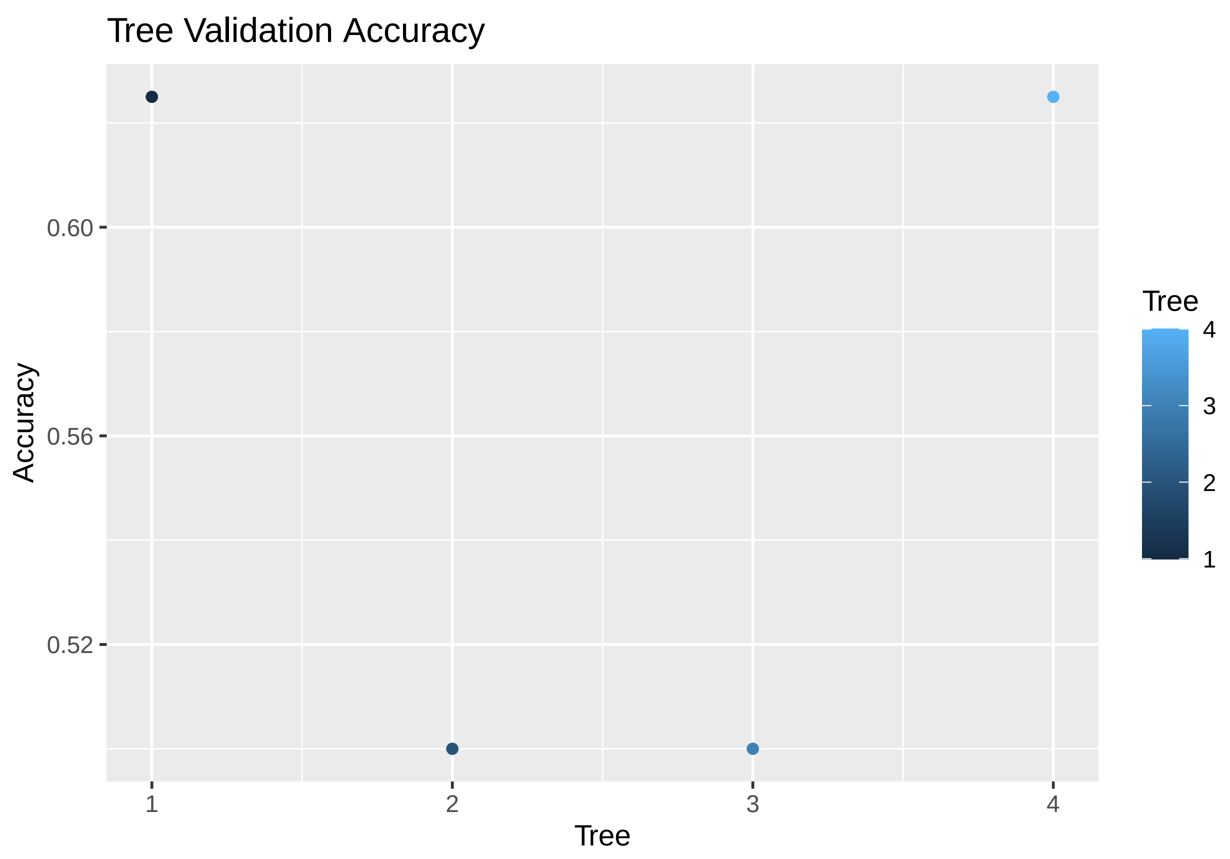
```
# predict
tree_pred1 = rpart.predict(skull_tree1, testing_df)
tree_pred2 = rpart.predict(skull_tree2, testing_df)
tree_pred3 = rpart.predict(skull_tree3, testing_df)
tree_pred4 = rpart.predict(skull_tree4, testing_df)

# create confusion matrix
confusion_matrix1 <- table(testing_df$Type,tree_pred1)
confusion_matrix2 <- table(testing_df$Type,tree_pred2)
confusion_matrix3 <- table(testing_df$Type,tree_pred3)
confusion_matrix4 <- table(testing_df$Type,tree_pred4)

# check accurary
accuracy1 <- sum(diag(confusion_matrix1))/sum(confusion_matrix1)
accuracy2 <- sum(diag(confusion_matrix2))/sum(confusion_matrix2)
accuracy3 <- sum(diag(confusion_matrix3))/sum(confusion_matrix3)
accuracy4 <- sum(diag(confusion_matrix4))/sum(confusion_matrix4)

# create dataframe
accuracy_df <- data.frame(Tree= c(1, 2, 3, 4)
                          , Accuracy = c(accuracy1
                                         , accuracy2
                                         , accuracy3
                                         , accuracy4))

# plot the accuracy
ggplot(accuracy_df) +
     geom_point(aes(x = Tree, y= Accuracy, color = Tree)) +
     ggtitle("Tree Validation Accuracy")
```



Trees 2 & 3 are clearly underfit with a 50% classification rate. Trees 1 and 4 are alittle better than a 50-50 guess so they may be useful for classification.

3.5)

Rpart using recursive binary paritioning to create trees. Specifically, rpart uses the gini index, to measure the impurity of node, combined with a loss function to create a generalized gini indexed that helps determine the split.

Problem 4

People largely ignore power analysis. i.e. the probability that they will obtain significant results. cohen gives an example of a study that resulted in accepting its null when there was only a 0.25 chance of rejecting it. This most likely due to lack of knowledge and complexity of material. So he is hoping this paper will remedy this.

I can say from my industry experience, power analysis was something that seemed illusive to me as well. I never came across it during my undergraduate education and it wasn't really the standard in the industry, I simply brushed off. I was also dealing with massive online experiments (sometimes sample sizes in the 100,000's) so I just assumed I had sufficient power.

He goes on to explain the relationship between alpha, N, effect size, and power. How each one is a function of the other three. The usual question being what N do i need to have X power for given alpha and ES.

Alpha is the risk of rejecting the null hypothesis when it is true. Power is the probablity of rejecting a false Ho.The effect size which is the the degree to which Ho is believed to be false.

While talking on ES, he proposed convention to make it easily understandable; small, medium, and large. He defines medium as “an effect likely to be visible by the naked eye of the careful observer.” I do not necessarily like this definition, as its incredibly vague and I think really is application dependent. He also says small is “noticeably smaller than medium but not so small to be trivial” Which seems incredibly subjective and isn’t really saying much. Although I understand what he is trying to do in making a quick reference page for anyone to use, so in those terms I would say at the very least its a good starting point.

4.2)

$$f = \sqrt{\frac{\eta^2}{1-\eta^2}}$$