Performance of a Casual Osu! Player 2.0

Yanyu Yang

4/21/2021

Introduction

My project will once again be focused on Osu!, a free PC rhythm game by Dean "peppy" Herbert. The main goal is to click circles in time to songs where users can create their own "map" of patterns.[1] Although I mainly play the game in single player mode, there is a built in ranking system. Performance points, officially abbreviated as "pp", are awarded at the end of every map. Rankings are calculated based on each player's total pp.

At the time of data collection on March 17, 2021, I was ranked 535,723rd out of the entire player base with 884pp. As one can deduct from the title, I am a fairly casual player. The following sections will analyze my Osu! performance.

Dataset

```
osu <- read.csv("~/sds 348/project/osu.csv")
head(osu)
```

```
##                             song   difficulty_name accuracy pp top_pp length
## 1           Koto no Ha (TV Size)       Gu's Insane    96.90 54     52     88
## 2                   &Z (TV size)            Insane    97.01 47     92     88
## 3                        unravel              Hard    91.70 45     44    203
## 4          BRAVE JEWEL (TV Size)             Hyper    95.05 44     72     89
## 5          IGNITE (TV size ver.)              Hard    99.06 44     52     87
## 6 Dancing stars on me! (TV Size) Chaozomi's Insane    87.81 44     51     97
##   bpm difficulty overweightness difficulty_category
## 1 145       3.91              0                Hard
## 2 158       4.20            169              Insane
## 3 135       3.52              0                Hard
## 4 192       3.98              3                Hard
## 5 171       3.43             12                Hard
## 6 137       3.67              0                Hard
```

The dataset I will use for this project is the merged data set from project 1. Some variables are taken from my Osu! profile, updated as of 3/17/2021. Other variables are taken from pps-by-grumd, a fanmade site which scrapes and collects official data for each Osu! map. The variable descriptions are listed below.

- **song**: the title of the song used for the map
- **difficulty_name**: the difficulty category, usually "easy", "hard", "insane", or "expert". difficulties listed as [name]'s [difficulty] indicate a guest mapper.

- **accuracy**: hits can get a score of 300, 100, 50, or miss. accuracy is calculated as (100*# of 300s) + (67.77*# of 100s) + (33.33* # of 50s) + (0* # of misses)
- **pp**: performance points awarded to me
- **top_pp**: average performance points awarded to the top 11,000 Osu! players. this more or represents the top pp ceiling, since even top players don't make perfect pp plays with 100% accuracy
- **length**: the length of the map in seconds
- **bpm**: beats per minute
- **difficulty**: the current difficulty system using stars. can range from 0 stars to 7+ stars
- **overweightness**: the number of top 11,000 players who have this map in their top 50 plays. higher overweightness means that map grants higher than average pp
- **difficulty_category**: difficulty by category

The last variable, `difficulty_category`, was converted from `difficulty` according to the Osu! Knowledge Base. The knowledge base, which is an official encyclopedia of Osu! terminology, lists the cutoffs for star ratings (`difficulty`) of each group (`difficulty_category`).

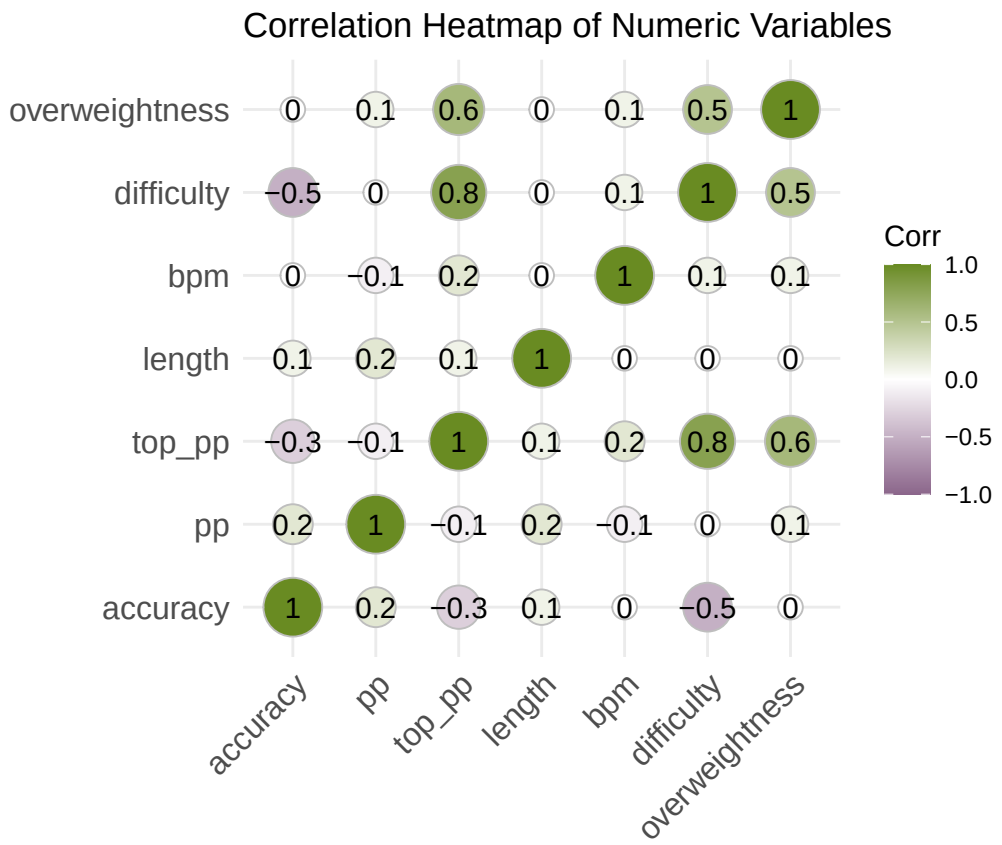
Tidy

Since I tidied the data set for my last project, I did not have to tidy it again. There are a total of 50 maps in this data set where each map is an individual observation, and a total of 10 variables. I am interested in doing further analysis on my Osu! performance using new tools we have learned in class. In particular, the main findings from project 1 showed a possible relationship between accuracy and pp across different category groups.

Exploratory Data Analysis

Correlation Heatmap

```
library(ggcorrplot)
library(tidyverse)
# make a dataset with only numeric variables
numeric <- osu %>%
  select_if(is.numeric)

# correlation matrix
corr <- round(cor(numeric), 1)

# correlation heat map
ggcorrplot(corr, method='circle',
           lab = TRUE,
           title = 'Correlation Heatmap of Numeric Variables',
           colors = c('plum4', 'white', 'olivedrab4'))
```



Considering Osu! is a circle clicking game, I made a correlation heatmap using circles. It seems that map difficulty ("difficulty") and average pp set by top 11,000 players ("top_pp") have the strongest positive correlation. On the other hand, map difficulty ("difficulty") and map accuracy set by me ("accuracy") have the strongest negative correlation.

This makes sense in context. Harder maps tend to give the highest pp, especially when played by the most competitive players. It is also pretty expected of me to have lowered accuracy when playing higher difficulty maps due to a lack of skill. In the following sections, I will further explore the relationships between the significant variables.

MANOVA

```
# manova
manova_osu <- manova(cbind(accuracy, pp, length, bpm) ~ difficulty_category,
                     data = osu)
summary(manova_osu)
```

```
##                      Df   Pillai approx F num Df den Df Pr(>F)
## difficulty_category   1 0.080897  0.99019      4     45 0.4226
## Residuals            48
```

Since all of the p values > 0.05, none of the response variables differ significantly by difficulty category.

ANOVA

I did not do ANOVA or post-hoc tests due to the fact that none of my response variables differed significantly by difficulty category.

Type 1 Error

```
# pr(at least 1 type 1 error)
1-(0.95^1)
```

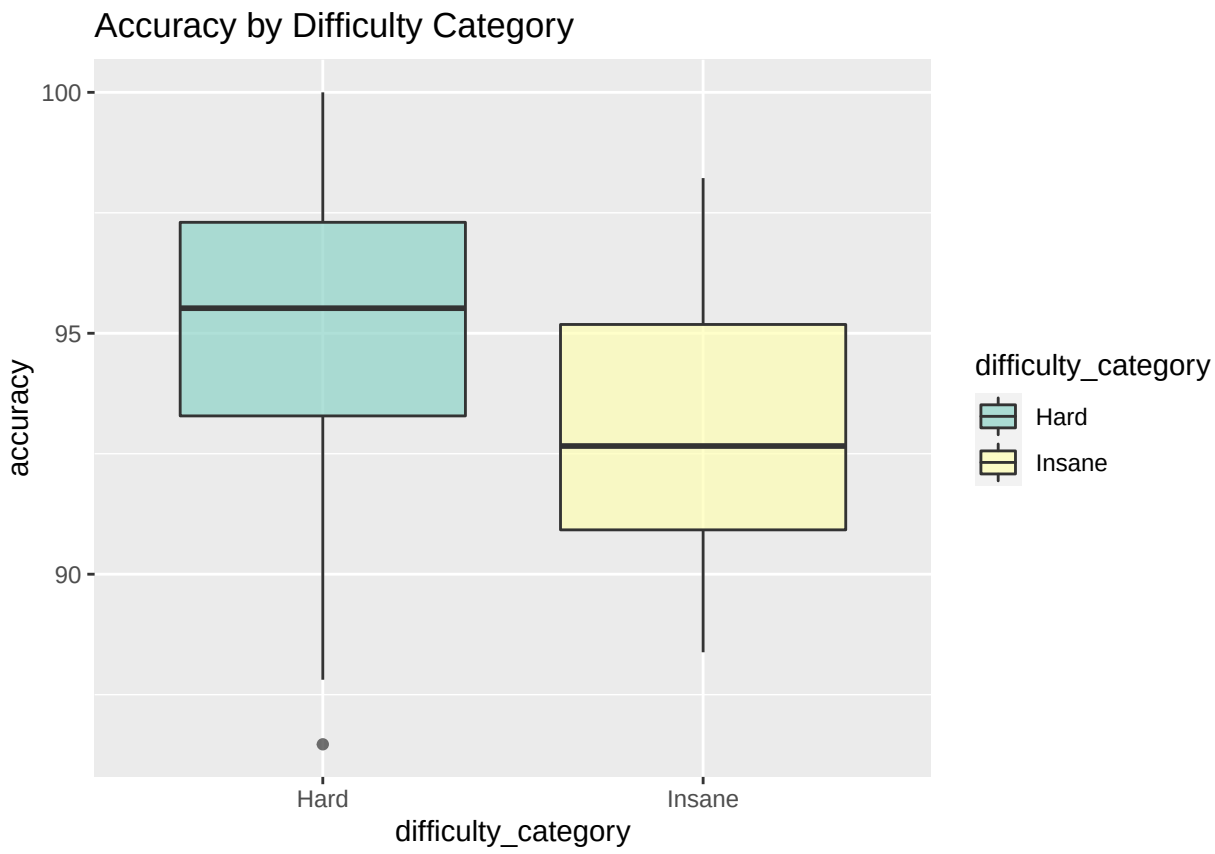
```
## [1] 0.05
```

I did 1 hypothesis test in the MANOVA. Therefore, my probability of making a type 1 error is 0.05.

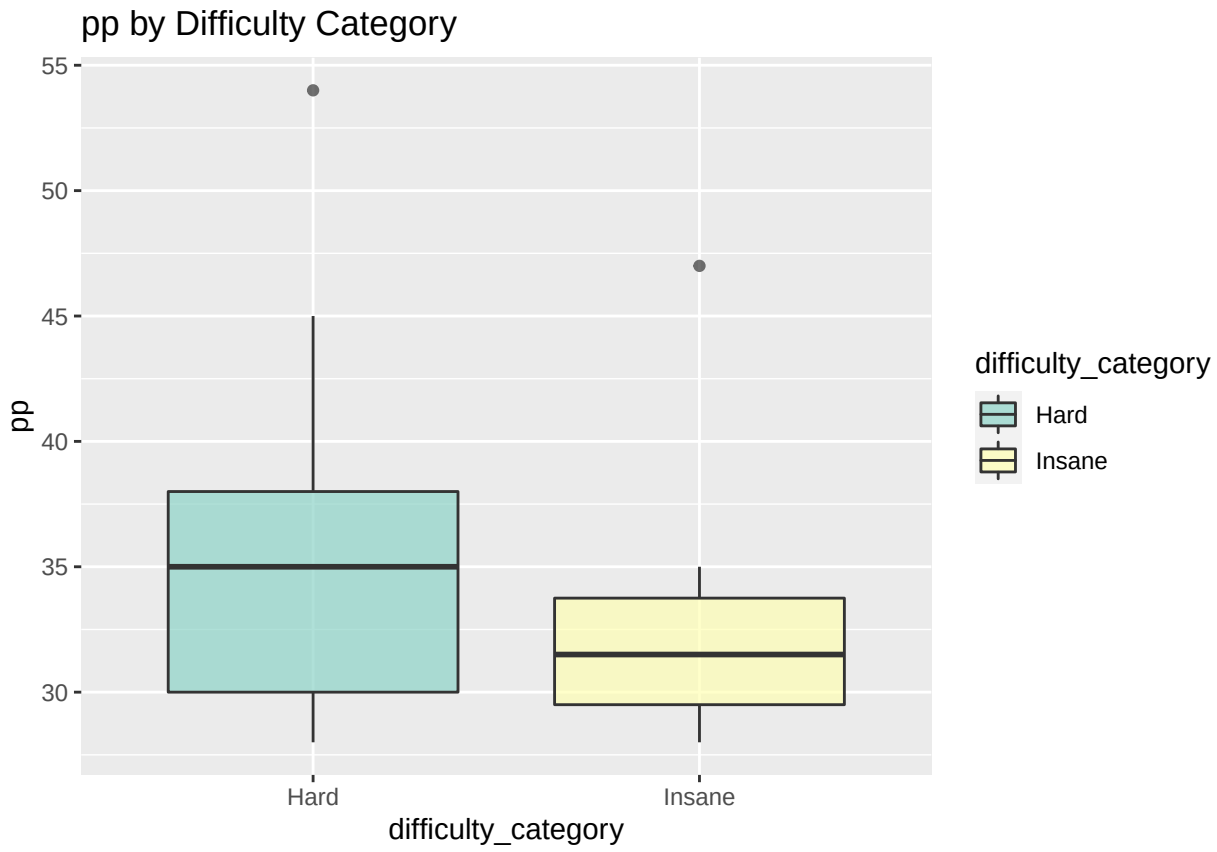
Assumptions

1. Sample size: 50 observations is much larger than the 5 variables I used, therefore sample size assumption is met.
2. Independence: All observations were individual plays by me.
3. Normality and Variance

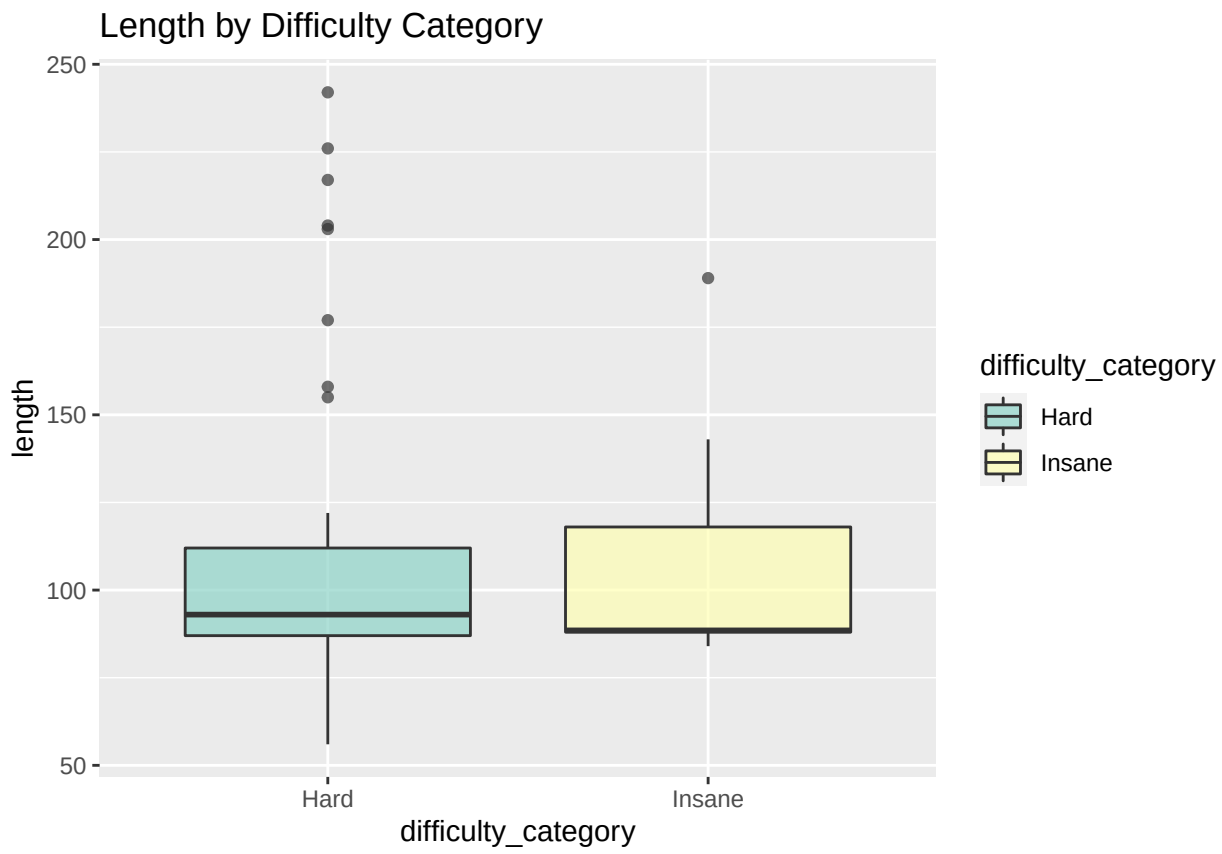
```
library(ggplot2)
# accuracy
ggplot(data = osu, aes(x = difficulty_category, y = accuracy, fill = difficulty_category)) +
  geom_boxplot(alpha = 0.7) +
  ggtitle('Accuracy by Difficulty Category') +
  scale_fill_brewer(palette = 'Set3')
```



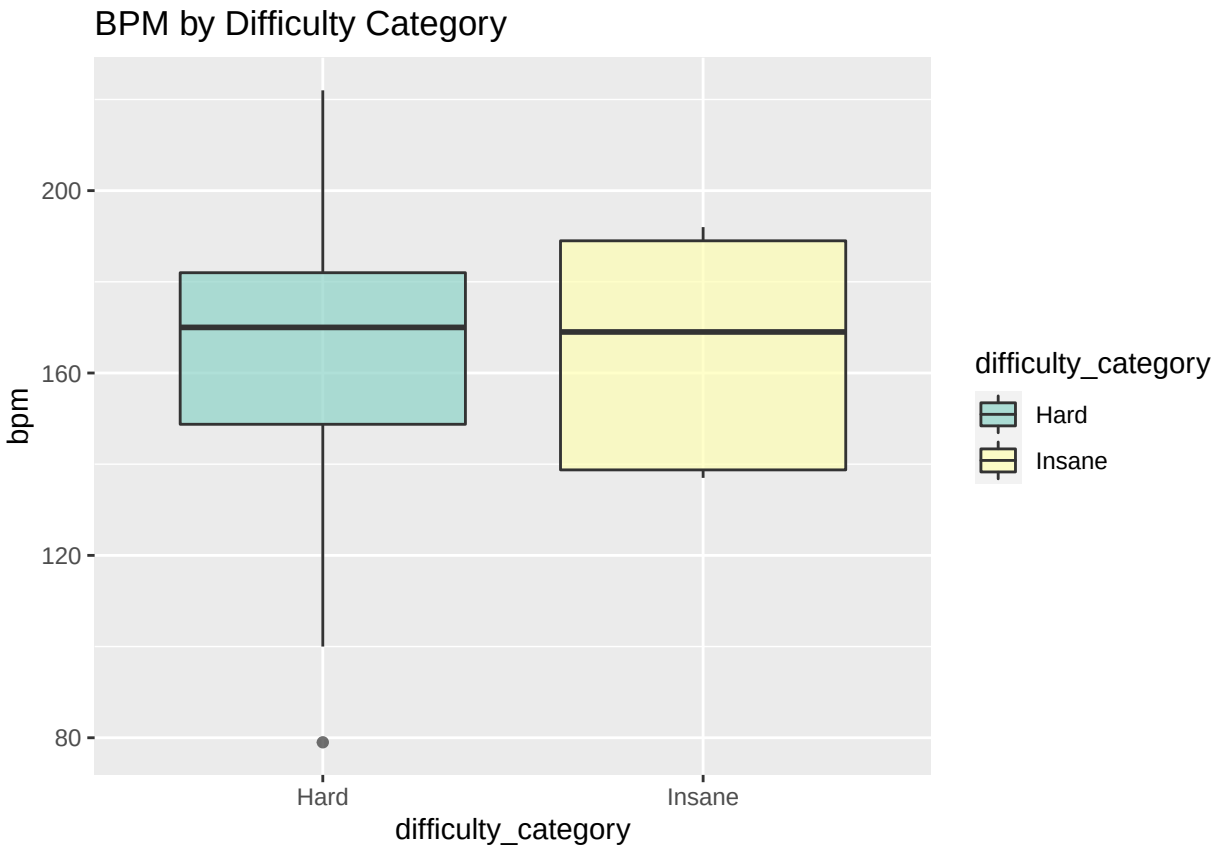

```
# pp
ggplot(data = osu, aes(x = difficulty_category, y = pp, fill = difficulty_category)) +
  geom_boxplot(alpha = 0.7) +
  ggtitle('pp by Difficulty Category') +
  scale_fill_brewer(palette = 'Set3')
```




```
# length
ggplot(data = osu, aes(x = difficulty_category, y = length, fill = difficulty_category)) +
  geom_boxplot(alpha = 0.7) +
  ggtitle('Length by Difficulty Category') +
  scale_fill_brewer(palette = 'Set3')
```




```
# bpm
ggplot(data = osu, aes(x = difficulty_category, y = bpm, fill = difficulty_category)) +
  geom_boxplot(alpha = 0.7) +
  ggtitle('BPM by Difficulty Category') +
  scale_fill_brewer(palette = 'Set3')
```



Generally speaking, accuracy, pp, and bpm were normally distributed across both categories, whereas length had significant right skews. Variance were also fairly similar across both categories, therefore the normality and variance assumption for MANOVA is mostly met.

Randomization Test

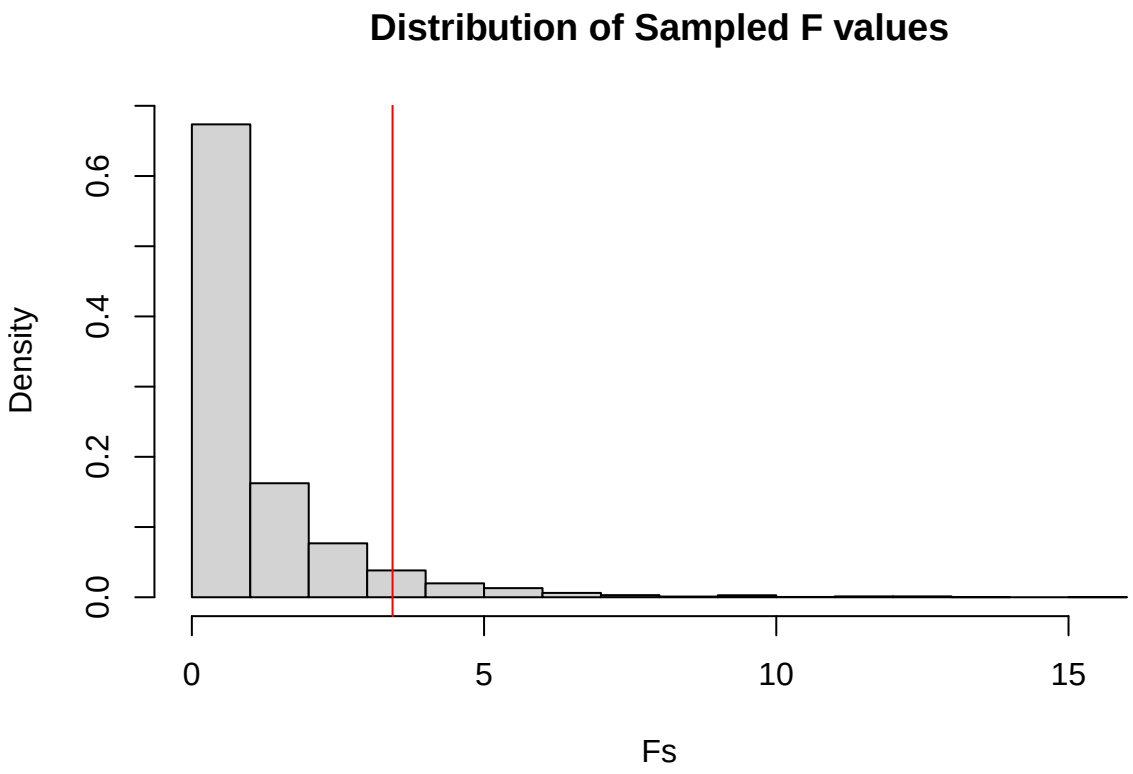
Although I did not find any significant effects in the MANOVA, I will be running an ANOVA to test the effects between accuracy and difficulty categories. My null hypothesis will be that there is no difference in accuracy between difficulty categories. My alternative hypothesis is that there is a significant difference in accuracy between difficulty categories.

```
# anova
summary(aov(accuracy ~ difficulty_category, data = osu))
```

```
##                    Df Sum Sq Mean Sq F value Pr(>F)
## difficulty_category  1   33.6   33.59   3.434   0.07 .
## Residuals           48  469.5    9.78
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

My observed F value is 3.434

```r
# observed F
obs_F <- 3.434

set.seed(348)
# randomization test
Fs <- replicate(5000,{
  # permute accuracy across difficulty categories
  new <- osu %>%
    mutate(accuracy = sample(accuracy))
  # within group variation
  SSW <- new %>%
    group_by(difficulty_category) %>%
    summarize(SSW = sum((accuracy - mean(accuracy))^2)) %>%
    summarize(sum(SSW)) %>%
    pull
  # between group variation
  SSB <- new %>%
    mutate(mean = mean(accuracy)) %>%
    group_by(difficulty_category) %>%
    mutate(groupmean = mean(accuracy)) %>%
    summarize(SSB = sum((mean - groupmean)^2)) %>%
    summarize(sum(SSB)) %>%
    pull
  # f statistic
  (SSB/1)/(SSW/48)
})

# Represent the distribution of the F-statistics for each randomized sample
hist(Fs, prob=T, main = 'Distribution of Sampled F values');
abline(v = obs_F, col="red",add=T)
```




```
# proportion of f statstic greater than F
mean(Fs > obs_F)
```

```
## [1] 0.0656
```

Only 6.56% of random samples have an F value greater or equal to the observed F value. We do not have strong enough support to reject the null hypothesis. In context, the null hypothesis means that my accuracy is about the same regardless of whether I'm playing a 'Hard' or 'Insane' map. This is good news for me, since it shows that I have pretty good accuracy across both difficulty categories!

Linear Regression Model

I will be predicting `accuracy` from `length` and `difficulty_category`. First, I will need to center length, which is a numeric variable. I will also code difficulty category as 0 if the map is 'Hard' and 1 if the map is 'Insane', using a new variable 'diff_code'.

```
# length center
osu <- osu %>%
  mutate(length_c = length - mean(length))

# code difficulty
osu <- osu %>%
  mutate(diff_code = ifelse(difficulty_category == "Hard", 0, 1))
```

I will then fit a regression model using the centered length and the coded difficulty variable.

```
model1 <- lm(accuracy ~ length_c + diff_code + length_c*diff_code, data = osu)
summary(model1)
```

```
##
## Call:
## lm(formula = accuracy ~ length_c + diff_code + length_c * diff_code,
##     data = osu)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6117 -2.0053  0.2318  2.1228  4.9174
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          9.508e+01  4.983e-01 190.818   <2e-16 ***
## length_c            -2.994e-05  1.067e-02  -0.003   0.9978
## diff_code           -1.936e+00  1.119e+00  -1.731   0.0902 .
## length_c:diff_code   3.461e-02  3.224e-02   1.074   0.2886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.151 on 46 degrees of freedom
## Multiple R-squared:  0.09226,    Adjusted R-squared:  0.03306
## F-statistic: 1.558 on 3 and 46 DF,  p-value: 0.2123
```

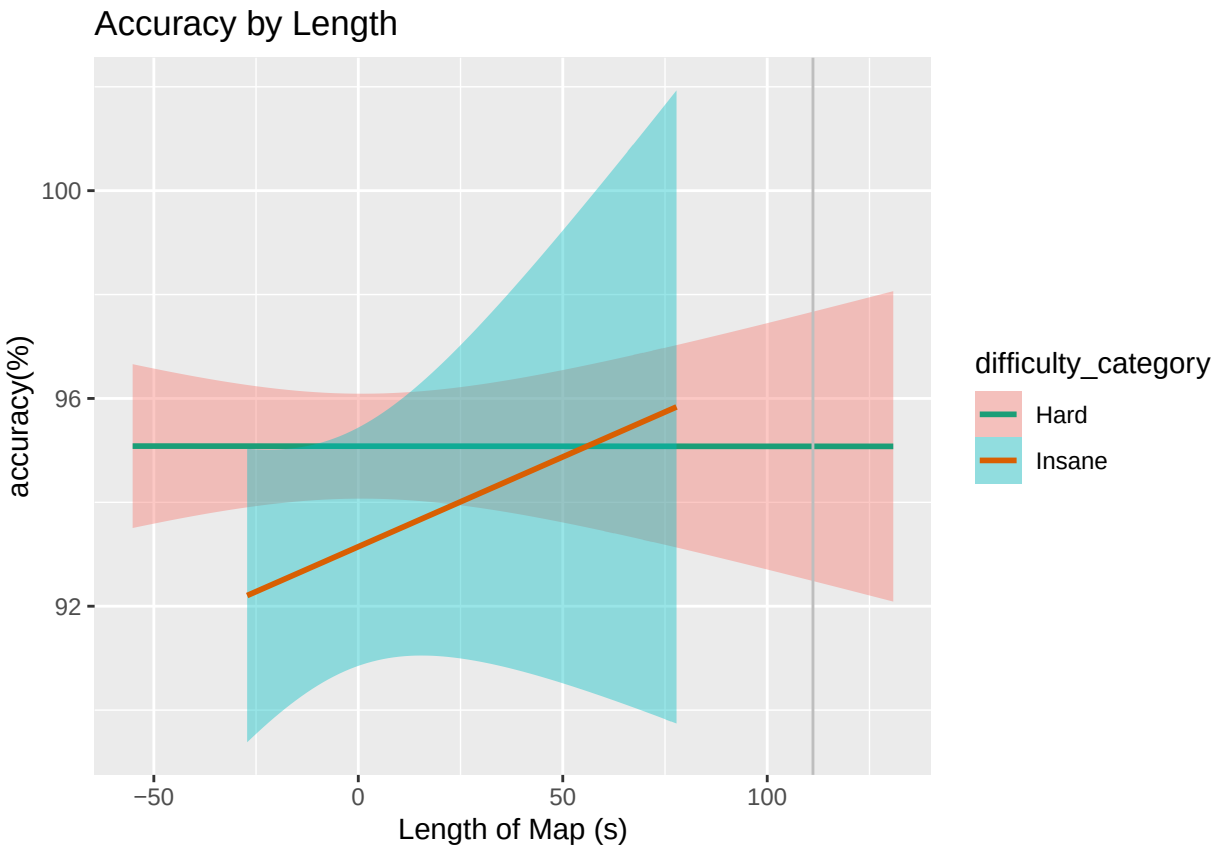
```
mean(osu$length)
```

```
## [1] 111.18
```

For every 1 second increase in map length from the mean of 111.18 seconds, accuracy decreases by 2.99e-05%. Insane maps have a lower accuracy of 1.94% compared to Hard maps. Holding map difficulty constant, an increase in map length of 1 second leads to an increase of accuracy of 0.035%.

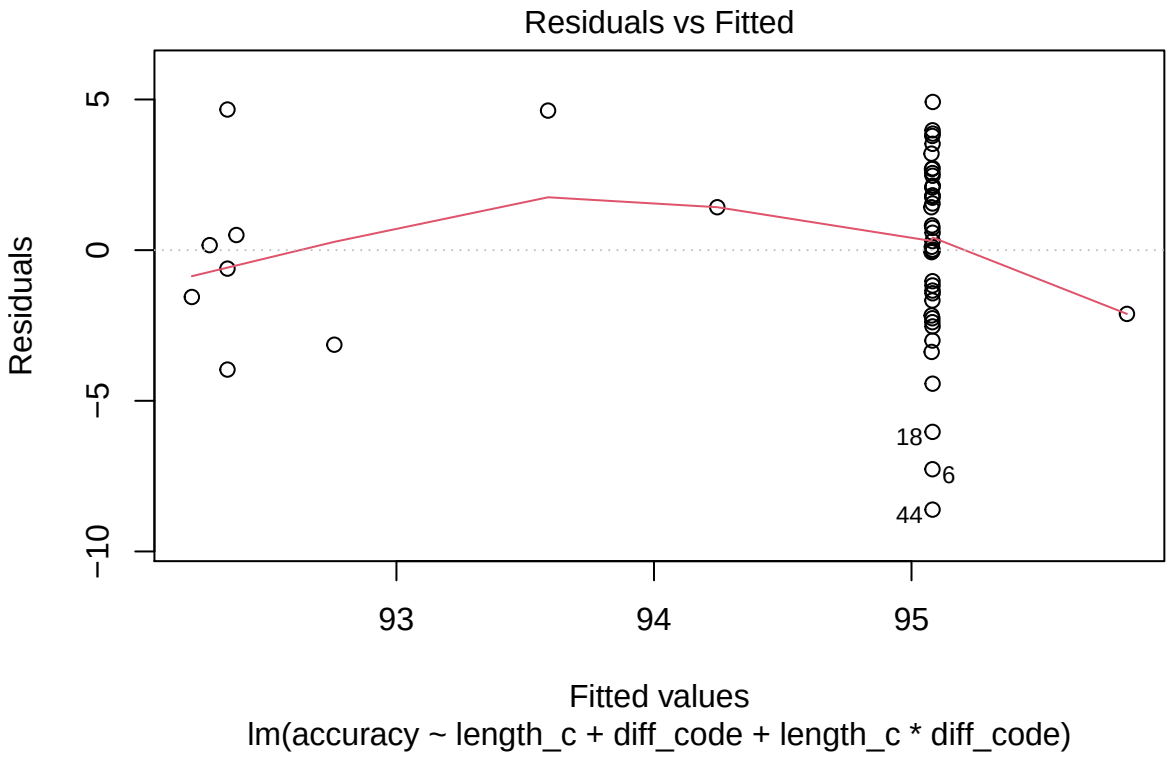
According to the adjusted R-squared value, this model explains 3.3% of the variation in accuracy. There are clearly many other variables in play.

```
# graph
ggplot(osu, aes(x = length_c, y = accuracy, fill = difficulty_category)) +
  geom_smooth(method = 'lm', aes(col = difficulty_category)) +
  ggtitle('Accuracy by Length') +
  scale_color_brewer(palette = 'Dark2') +
  geom_vline(xintercept = mean(osu$length, na.rm = TRUE), col='gray') +
  labs(x = 'Length of Map (s)',
       y = 'accuracy(%)')
```

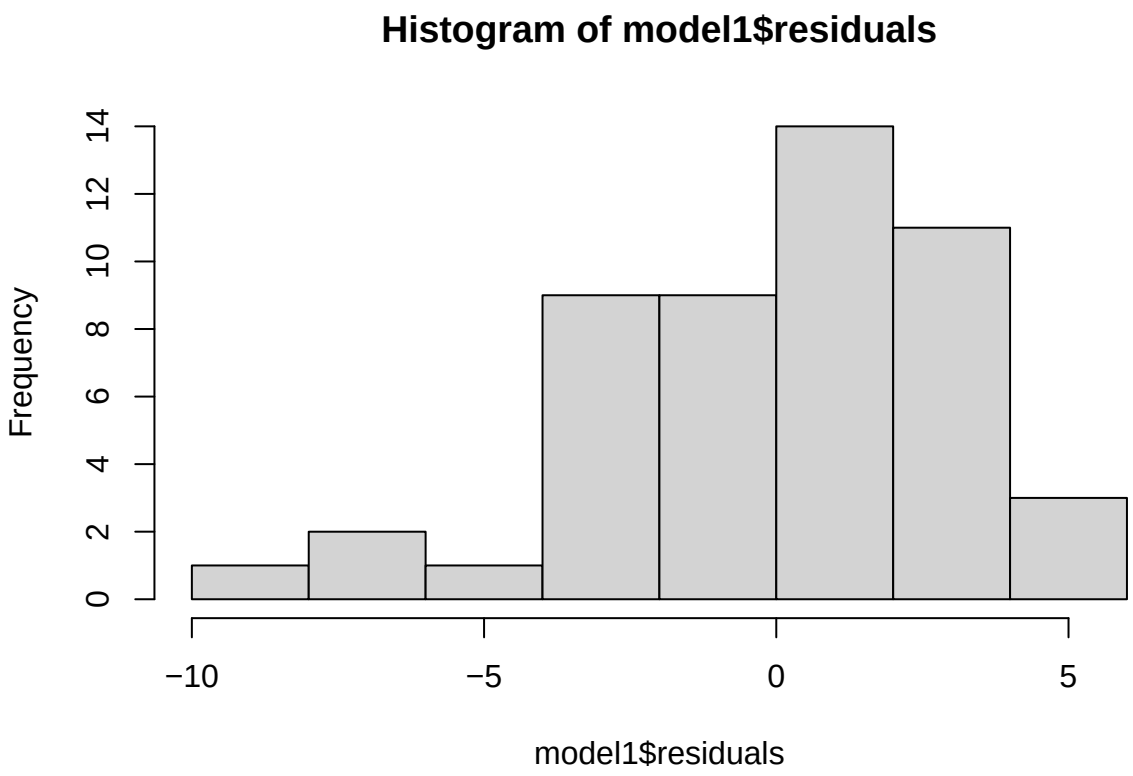


Assumptions

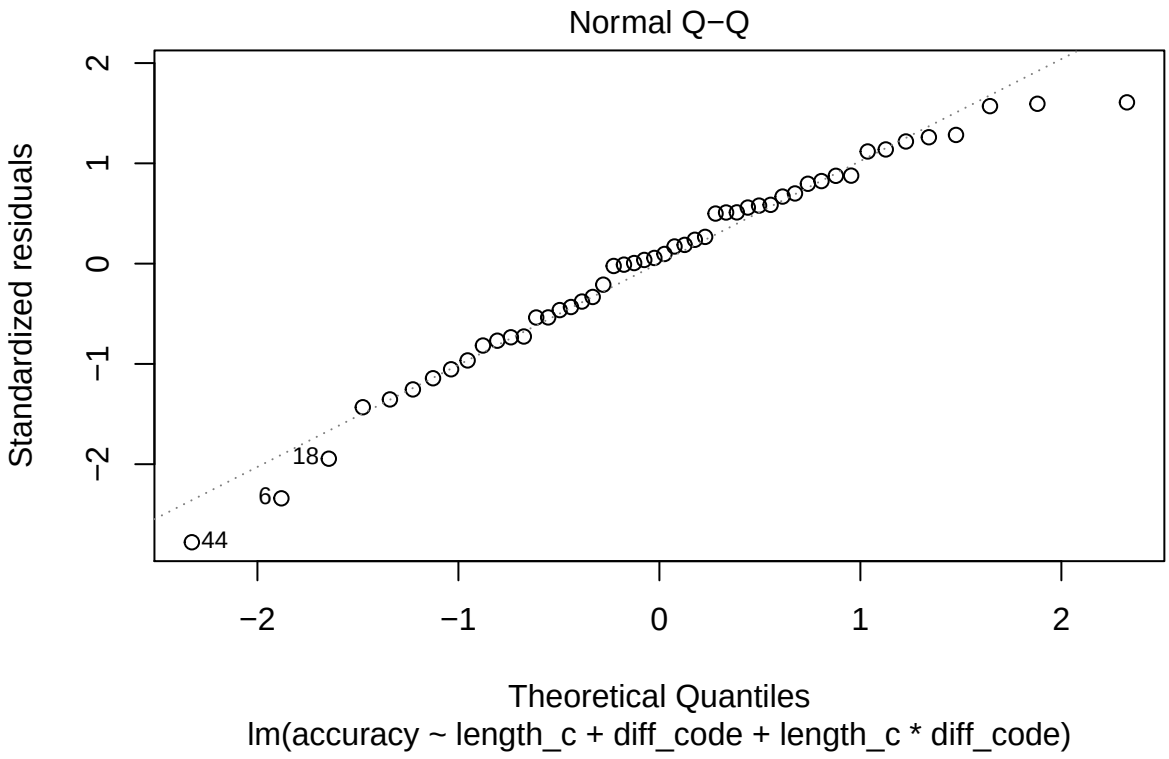
```
# residuals against fitted values
plot(model1, which = 1)
```




```
# histogram of residuals
hist(model1$residuals)
```




```r
# qq plot
plot(model1, which = 2)
```




```
# normality
shapiro.test(model1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model1$residuals
## W = 0.96568, p-value = 0.1536
```

```
#homoscedasity
library(sandwich)
library(lmtest)

# Breusch-Pagan test
# H0: homoscedasticity
bptest(model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model1
## BP = 1.5716, df = 3, p-value = 0.6658
```

The residual plot shows a slight curve in the red line, but over all there does not appear to be funneling or unequal variances. If anything, it seems to show that most of my observations have around a 95% accuracy. The normality assumption is met according to the QQ plot and the Shapiro-Wilk normality test ($p = 0.15$).

Robust Standard Errors

```
# uncorrected
summary(model1)$coef
```

```
##                          Estimate Std. Error       t value     Pr(>|t|)
## (Intercept)         9.508102e+01 0.49828019 190.818392816 2.458963e-68
## length_c           -2.993943e-05 0.01067181  -0.002805469 9.977737e-01
## diff_code          -1.935594e+00 1.11851099  -1.730509831 9.024352e-02
## length_c:diff_code  3.461234e-02 0.03224028   1.073574515 2.886131e-01
```

```
# robust
coeftest(model1, vcov = vcovHC(model1))
```

```
##
## t test of coefficients:
##
##                        Estimate  Std. Error  t value Pr(>|t|)
## (Intercept)         9.5081e+01  5.1096e-01 186.0830   <2e-16 ***
## length_c           -2.9939e-05  9.7886e-03  -0.0031   0.9976
## diff_code          -1.9356e+00  1.4336e+00  -1.3502   0.1836
## length_c:diff_code  3.4612e-02  5.9210e-02   0.5846   0.5617
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After calculating the robust standard errors, none of the slopes are significant in the new regression model. Previously, the slope for difficulty category and the interaction of length and difficulty category were significant.

Bootstrap

```
set.seed(348)
samp_SEs <- replicate(5000, {
  boot_data <- sample_frac(osu, replace = TRUE)
  # fit regression model
  fitboot <- lm(accuracy ~ length_c + diff_code + length_c*diff_code, data = osu)
  # save the coefficients
  coef(fitboot)
})

# estimated SEs
samp_SEs %>%
  t %>%
  as.data.frame %>%
  summarize_all(sd)
```

```
##   (Intercept) length_c diff_code length_c:diff_code
## 1           0        0         0                  0
```

```r
# confidence interval for the estimates
samp_SEs %>%
  t %>%
  as.data.frame %>%
  pivot_longer(everything(), names_to = "estimates", values_to = "value") %>%
  group_by(estimates) %>%
  summarize(lower = quantile(value,.025), upper = quantile(value,.975))
```

```
## # A tibble: 4 x 3
##   estimates               lower      upper
## * <chr>                   <dbl>      <dbl>
## 1 (Intercept)        95.1       95.1
## 2 diff_code          -1.94      -1.94
## 3 length_c           -0.0000299 -0.0000299
## 4 length_c:diff_code  0.0346     0.0346
```

```r
# compare with normal-theory SEs
coeftest(model1)[,1:2]
```

```
##                          Estimate Std. Error
## (Intercept)         9.508102e+01 0.49828019
## length_c           -2.993943e-05 0.01067181
## diff_code          -1.935594e+00 1.11851099
## length_c:diff_code  3.461234e-02 0.03224028
```

```r
# compare with robust SEs
coeftest(model1, vcov = vcovHC(model1))[,1:2]
```

```
##                          Estimate Std. Error
## (Intercept)         9.508102e+01 0.51096036
## length_c           -2.993943e-05 0.00978859
## diff_code          -1.935594e+00 1.43358883
## length_c:diff_code  3.461234e-02 0.05921032
```

All 4 coefficients appear to be very similar to what we had before, both in regression with uncorrected standard error and the regression with the robust standard error.

Logistic Regression

For the logistic regression, I will be using the previously created `diff_code` variable as my binary categorical variable. 0 represents 'Hard' while 1 represents 'Insane'. My predictor variables will be accuracy and pp.

```r
# new regression model
model2 <- glm(diff_code ~ accuracy + pp, data = osu, family = binomial(link="logit"))
summary(model2)
```

```
##
## Call:
## glm(formula = diff_code ~ accuracy + pp, family = binomial(link = "logit"),
##     data = osu)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3189  -0.6746  -0.5354  -0.3702   2.3869
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 17.84842   10.82865   1.648   0.0993 .
## accuracy    -0.17992    0.11618  -1.549   0.1215
## pp          -0.06774    0.07631  -0.888   0.3747
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 50.040  on 49  degrees of freedom
## Residual deviance: 45.936  on 47  degrees of freedom
## AIC: 51.936
##
## Number of Fisher Scoring iterations: 4
```

In this context, accuracy decreases by 0.19% when the map is of 'Insane' difficulty. Pp also decreases by 0.068 units when the map is 'Insane' compared to 'Hard'.

Confusion Matrix

```
# predicted probabilities
osu$prob <- predict(model2, type = "response")

# if the probability is greater than 0.5, the map is 'Insane'
osu$predicted <- ifelse(osu$prob > .5, "Insane", "Hard")

# confusion matrix
table(truth = osu$diff_code, prediction = osu$predicted)
```

```
##      prediction
## truth Hard Insane
##     0   39      1
##     1    9      1
```

```
# accuracy
(39+1)/50
```

```
## [1] 0.8
```

```
# sensitivity
1/10
```

```
## [1] 0.1
```

```
# specificity
39/40
```

```
## [1] 0.975
```

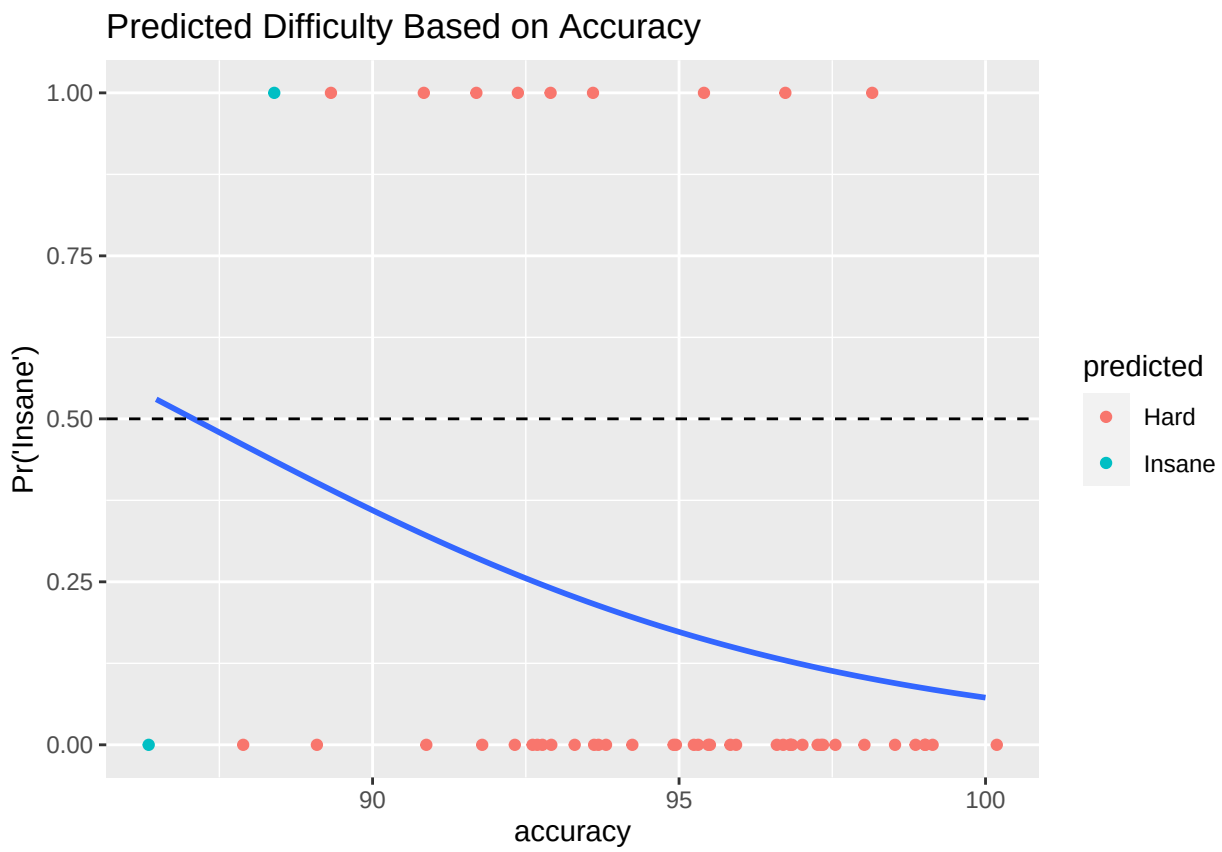
```
# precision
1/2
```

```
## [1] 0.5
```

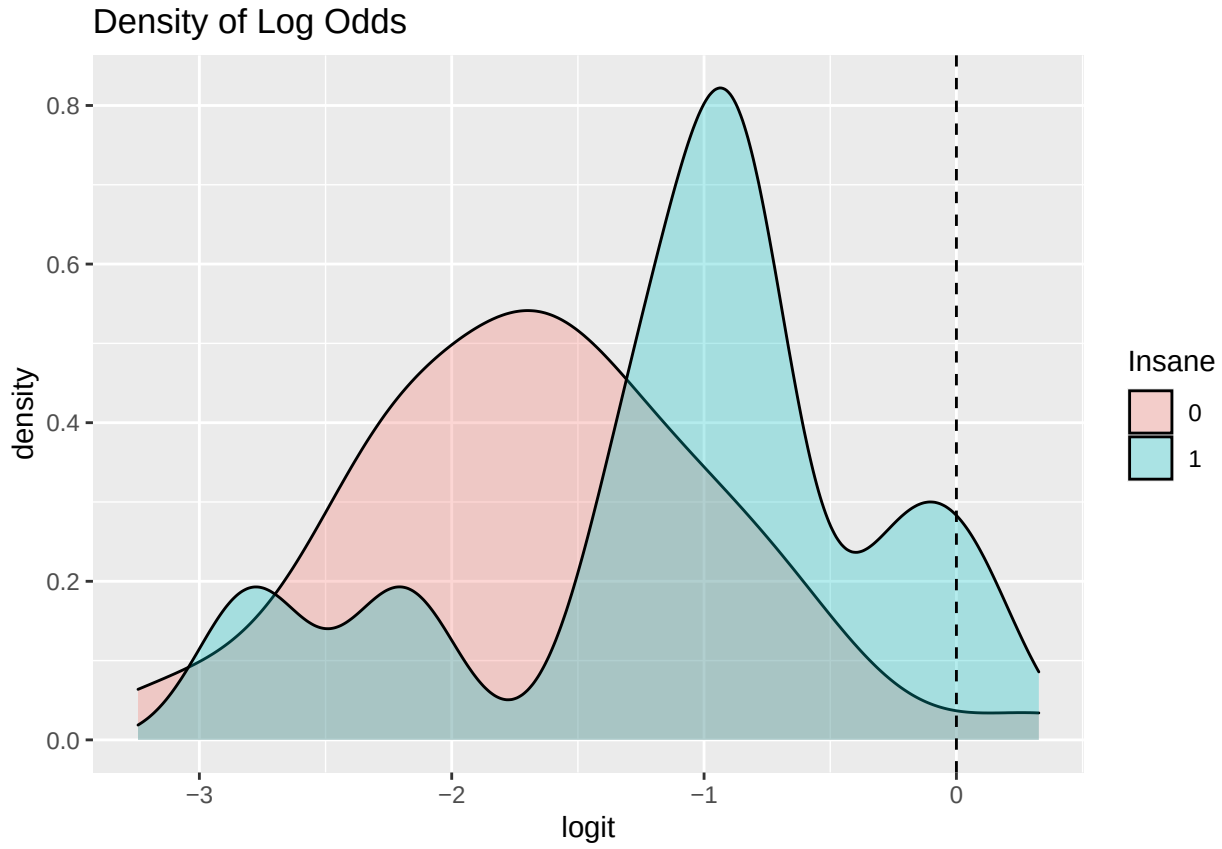
The accuracy is 0.8, the sensitivity is 0.1, the specificity is 0.975, and the precision is 0.5.

Density Model

```
# plot model
ggplot(osu, aes(accuracy, diff_code)) +
  geom_jitter(aes(color = predicted), width = .3, height = 0) +
  stat_smooth(method="glm", method.args = list(family="binomial"), se = FALSE) +
  geom_hline(yintercept = 0.5, lty = 2) +
  ylab("Pr('Insane')") +
  ggtitle('Predicted Difficulty Based on Accuracy')
```

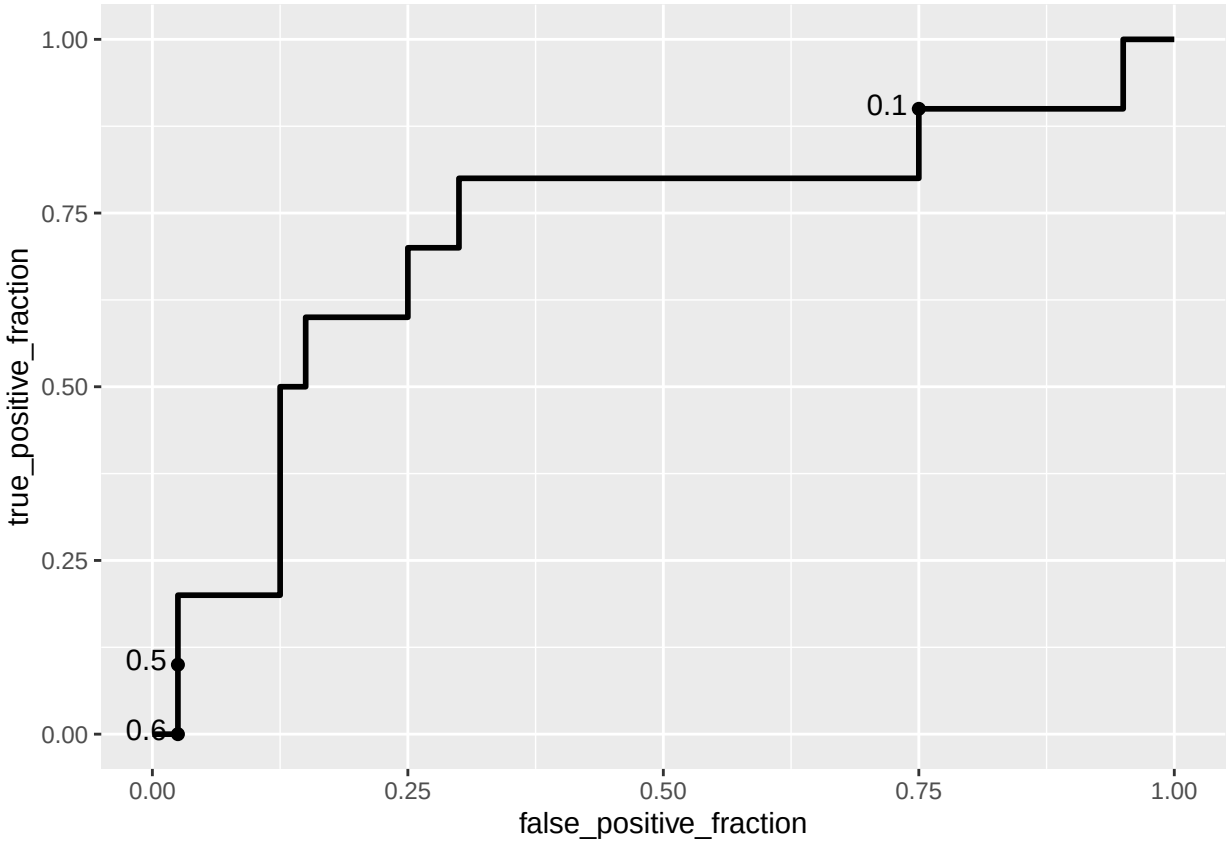


```
# save the predicted log-odds in the dataset
osu$logit <- predict(model2)

# compare to the outcome in the dataset with a density plot
ggplot(osu, aes(logit, fill = as.factor(diff_code))) +
  geom_density(alpha = .3) +
  geom_vline(xintercept = 0, lty = 2) +
  labs(fill = "Insane") +
  ggtitle('Density of Log Odds')
```



ROC and AUC

```
library(plotROC)

# plot ROC
ROCplot1 <- ggplot(osu) +
  geom_roc(aes(d = diff_code, m = prob), cutoffs.at = list(0.1, 0.5, 0.9))
ROCplot1
```




```
# AUC
calc_auc(ROCplot1)
```

```
##   PANEL group    AUC
## 1     1    -1 0.7175
```

The area under the ROC curve is 0.7175. Using the rule of thumb, the model has a fair prediction power. This means that I can predict somewhat accurately whether a map is 'Hard' or 'Insane' based on accuracy.

References

[1] FAQ. (n.d.). Retrieved from https://osu.ppy.sh/wiki/en/FAQ [2] Pps by grumd - osu! farm maps. (n.d.). Retrieved from https://osu-pps.com/#/osu/maps [3] Slendermaniac · player info: Osu! (n.d.). Retrieved from https://osu.ppy.sh/users/4655284