Project 2

Kali, Blake, Daniel, and Tony

12/2/20

Introduction

```
For our second project, we decided to use the same data set that we used in our first
project which contains key data points collected from various matches in the game Lea
gue of Legends. League of Legends is a multiplayer online Battle Arena (MOBA) in whic
h two teams of five players each compete to destroy the opposing team’s base, called
the Nexus.

The nexus of each team spawns small, NPC characters called minions (sometimes creeps)
that follow one of three lanes and support the player(s) that has chosen that lane to
capture objectives and move closer towards the enemy nexus. Each lane is guarded by t
hree towers, each at different locations. These towers also support the player by def
eating minions and enemy champions who happen to be in the tower’s range. Once the th
ree towers have been destroyed, the team can move on to destroy the enemy’s inhibito
r. This inhibitor prevents the opposing team from summoning an upgraded version of th
ese minions, who deal increased damage and have significantly more health. Each team
has three inhibitors, each one corresponding to a lane. Only one needs to be destroye
d before the next stage can be completed. The final stage consists of two towers prot
ecting the nexus. Both towers and at least one inhibitor must be destroyed in order t
o begin damaging the Nexus. The team who destroys their opponent’s Nexus first wins t
he game.


The teams are divided into sub roles that players can choose, which dictates the lane
that their champion plays in. Each player must choose a champion character to control
for the course of the game, and each champion has unique abilities and play styles to
fill the roles that each team must have. Some champions excel full frontal encounter
s, while others are more suited towards stealth and quick strikes, while few can heal
and protect their team. Players kill both enemy minions, elite neutral monsters, and
enemy players to increase the power of their champion and buy items from the gold tha
t these kills produce. Items increase the stats of the player and allow for the playe
r to “build” into a sort of play style that is suited for the champion they chose.

In this project, we will attempt to determine what variables in this set are best at
predicting a win in League of Legends.
```

Import + Cleaning

```r
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✔ ggplot2 3.3.2     ✔ purrr   0.3.4
## ✔ tibble  3.0.3     ✔ dplyr   1.0.2
## ✔ tidyr   1.1.2     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
data <- read.csv("https://raw.githubusercontent.com/kalihale/miscfiles/main/high_diam
ond_ranked_10min.csv")
names(data)[names(data) == "blueWins"] <- "Wins"
data$Wins <- as.factor(data$Wins)
levels(data$Wins) <- c("red", "blue")
data$blueFirstBlood <- as.factor(data$blueFirstBlood)
levels(data$blueFirstBlood) <- c("FALSE", "TRUE")
data$redFirstBlood <- as.factor(data$redFirstBlood)
levels(data$redFirstBlood) <- c("FALSE", "TRUE")

dataBlueWins <- data[data$Wins == "blue", ]
dataRedWins <- data[data$Wins == "red", ]
```

Data Dictionary

```
This data set is a collection of many of the metrics listed above, and will be used i
n our analysis to see which of these metrics seems to contribute the most to a victor
y for a given team.For the sake of our analysis, we have set up a dictionary to make
our data more understandable. The following is a list of terms and how they are to be
understood within the Data Set.
```

- “Red” – The Color of the Opposing Team
- “Blue” – The Color of the Current Team
- gameId – The Riot ID assigned to the match
- Champion – Refers to the playable character that the player has chosen to compete with.
- Ward – Device used to grant players vision of areas that they are not currently in range to see.
- First Blood - Refers to the first player kill of the game.
- Kill – Refers to the act of a player defeating another player through combat.
- Death – Refers to the act of being defeated in both Player vs. Player and environmental combat.
- Assists – Refers to score points given to teammates who help another player in killing an enemy.
- Lane – Refers to one of the three standard areas in which players compete (Top, Middle, Bottom). They

dictate where champions are best suited as well as where minions can be killed.

- Jungle - Refers to the area outside of the lanes where additional monsters exist that can be slain, including large and elite monsters.
- Minions / Creeps – Small non-player characters who roam the lanes of the game, attempting to help players reach the goal of destroying the enemy Nexus.
- Monster – Small non-player characters who reside in the Jungle and are used by champions to gather resources and experience.
- Elite / Elite Monsters – Large,non-player characters who are substantially tougher than the regular monsters and tend to provide some bonus to the player after being slain.
- Dragon – Large monster that resides in the bottom half of the jungle,that comes in four variants. Each variant corresponds to an element, being Earth, Fire, Water, and Air. Dragons provide a bonus depending on the element when killed. When a team kill four dragons, they receive a large bonus.
- Herald – Large monster that resides in the top half of the jungle, when killed drops and eye which can be used by the player to summon a friendly version of the herald that will attack enemy towers and deal massive amounts of damage.
- Baron / Baron Nashor – Large Elite Monster that resides in the top half of the jungle. Typically require a full team effort to kill. When defeated, Baron drops a buff to the team which killed him that increases the effectiveness that minions have.
- Gold - Currency collected in the game used to buy equipment for champions.
- Experience - Points used to gauge progress towards champion levels, awarded by killing players, monsters, and minions.
- Level – Refers to the current tier that the chosen champion currently resides as. Each level comes with increased value of that champion.
- CS / Creep Score / MS / Minion Score – Score that each player has which is a sum tally of how many non-playable enemies that the player has killed.
- Diff – Refers to the difference in between two measurements.

Data Analysis

```
In this section, we are going to explore if the elite monsters on the map will affect
the winrate. Since the dragon spawn at 5:00 and respawn every 5 minutes after killed,
the Herald spawns at 8:00 and respawns once 6 minutes after being killed, there could
only be 1 being capture for the first 10 minutes of a game.
```

```
#Make data$Wins numerical to calculate the winrate
levels(data$Wins) <- c("0", "1")
data$Wins <- as.numeric(data$Wins)
##Select Data
Blue_dragon <- data[data$blueDragons==1,]
Red_dragon <- data[data$redDragons==1,]

Blue_Herald <- data[data$blueHeralds==1,]
Red_Herald <- data[data$redHeralds==1,]

Blue_both <- data[data$blueEliteMonsters==2,]
Red_both <- data[data$redEliteMonsters==2,]

nrow(Blue_dragon)/nrow(data)
```

```
## [1] 0.36198
```

```
nrow(Red_dragon)/nrow(data)
```

```
## [1] 0.4130985
```

```
##Red side have slightly more chance to get a dragon
nrow(Blue_Herald)/nrow(data)
```

```
## [1] 0.1879745
```

```
nrow(Red_Herald)/nrow(data)
```

```
## [1] 0.1600364
```

```
##Blue side have slightly more chance to get a herald
nrow(Blue_both)/nrow(data)
```

```
## [1] 0.07186962
```

```
nrow(Red_both)/nrow(data)
```

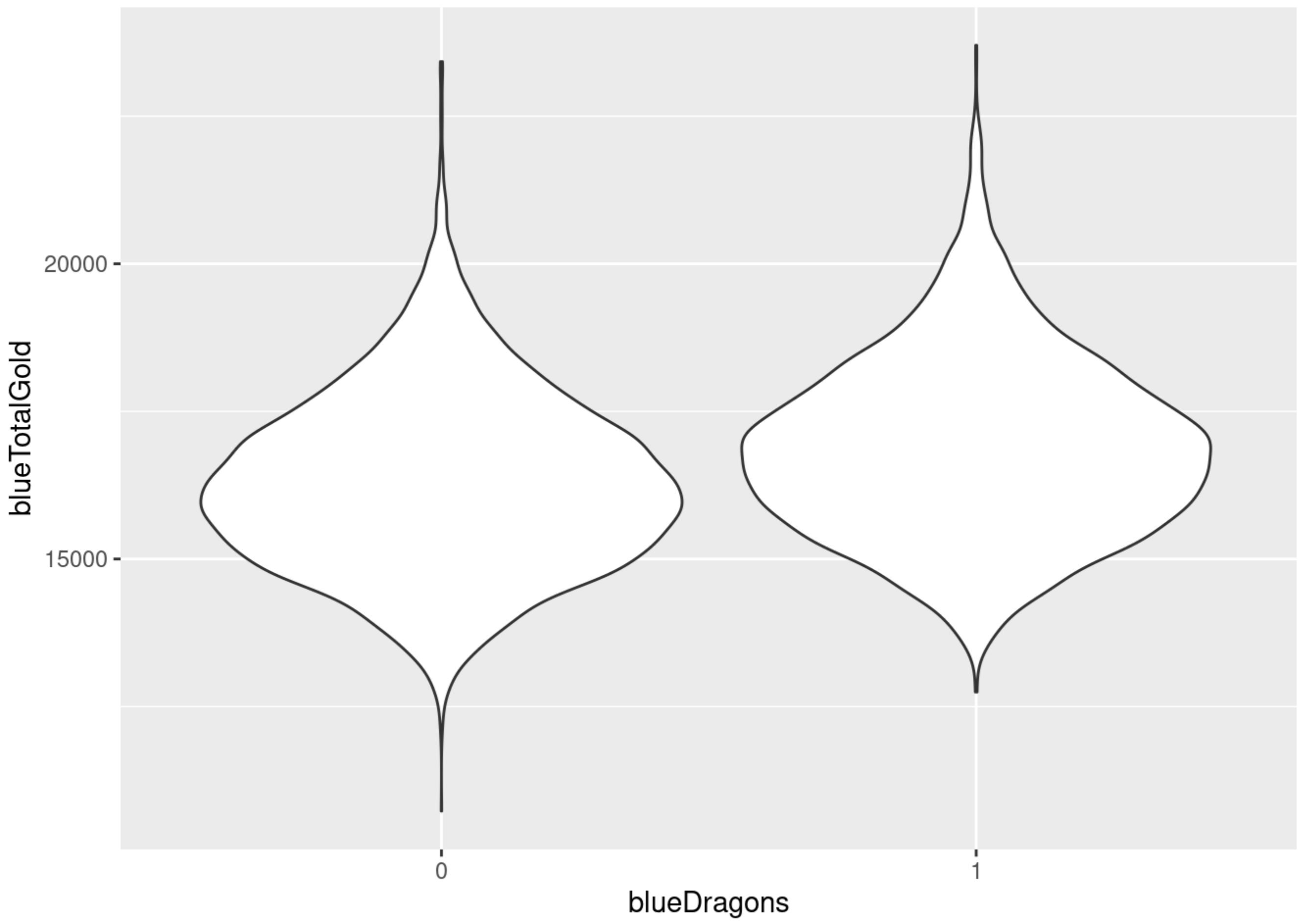
```
## [1] 0.07389412
```

```
##About the same
```

Does dragon make a difference in winrate and gold?

```
H0: No. The mean difference of winrate is less than or equals to 0.
Ha: Yes. The mean difference of winrate is greater than 0.
```

```
data$blueDragons <- as.factor(data$blueDragons)
data$redDragons <- as.factor(data$redDragons)
data$blueHeralds <- as.factor(data$blueHeralds)
data$redHeralds <- as.factor(data$redHeralds)
ggplot(data = data, aes(x = blueDragons, y = blueTotalGold)) + geom_violin()
```



```
t.test(data$Wins,Blue_dragon$Wins)$p.value
```

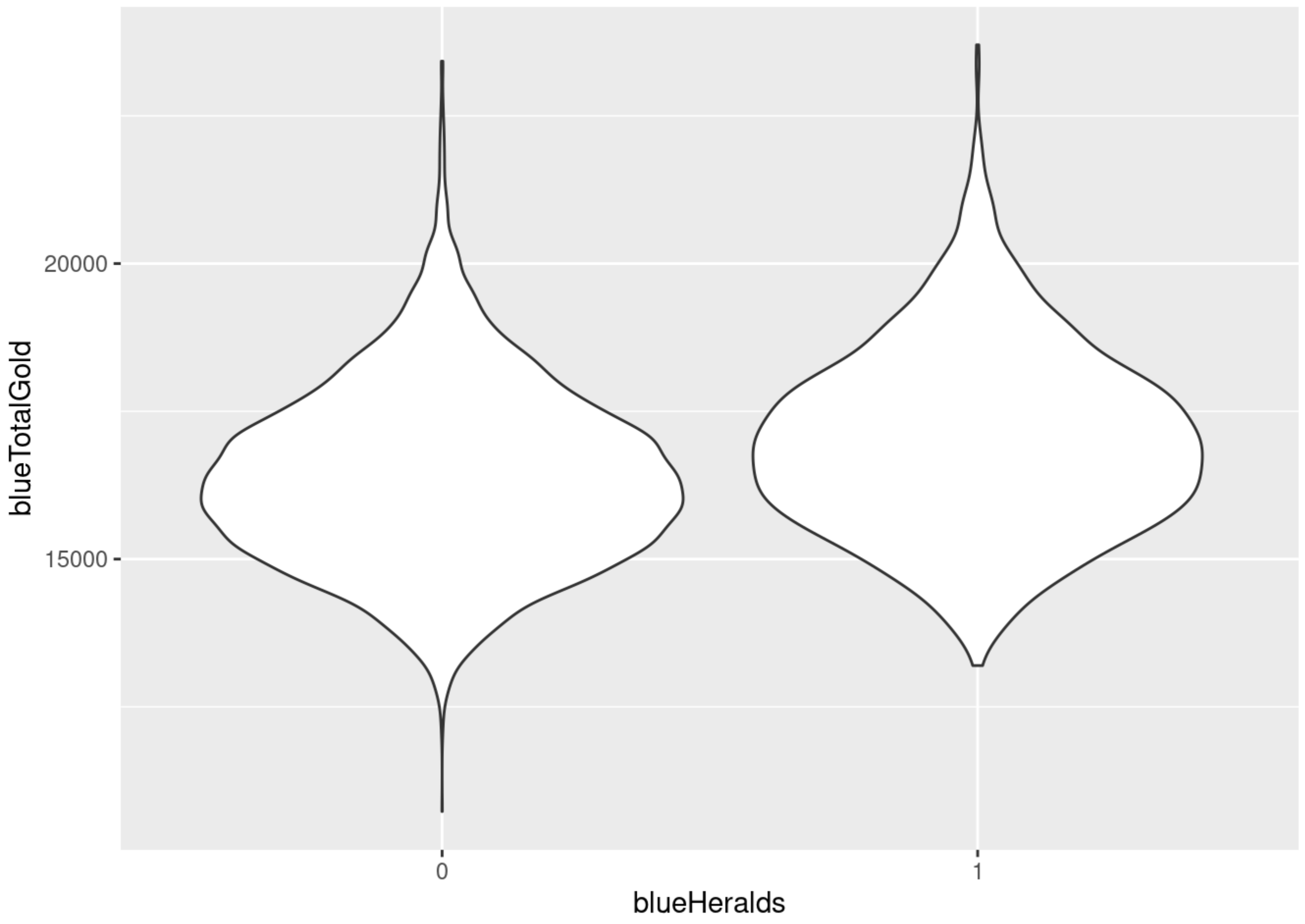
```
## [1] 6.181448e-50
```

```
Very small number. We can reject the null hypothesis. We can conclude that getting dr
agon can definitly increase the winrate and, from our plot, it appears to also increa
se the gold of the team that gets it.
```

Does Herald make a difference in winrate and gold?

```
H0: No. The mean difference of winrate is less than or equals to 0.
Ha: Yes. The mean difference of winrate is greater than 0.
```

```
ggplot(data = data, aes(x = blueHeralds, y = blueTotalGold)) + geom_violin()
```



```
t.test(data$Wins,Blue_Herald$Wins)$p.value
```

```
## [1] 1.797869e-14
```

```
Very small number. We can reject the null hypothesis. We can conclude that getting he
rald can definitly increase the winrate and, from our plot, it appears to also increa
se the gold of the team that gets it.
```

How about both?

```
H0: No. The mean difference of winrate is less than or equals to 0.
Ha: Yes. The mean difference of winrate is greater than 0.
```

```
t.test(data$Wins,Blue_both$Wins)$p.value
```

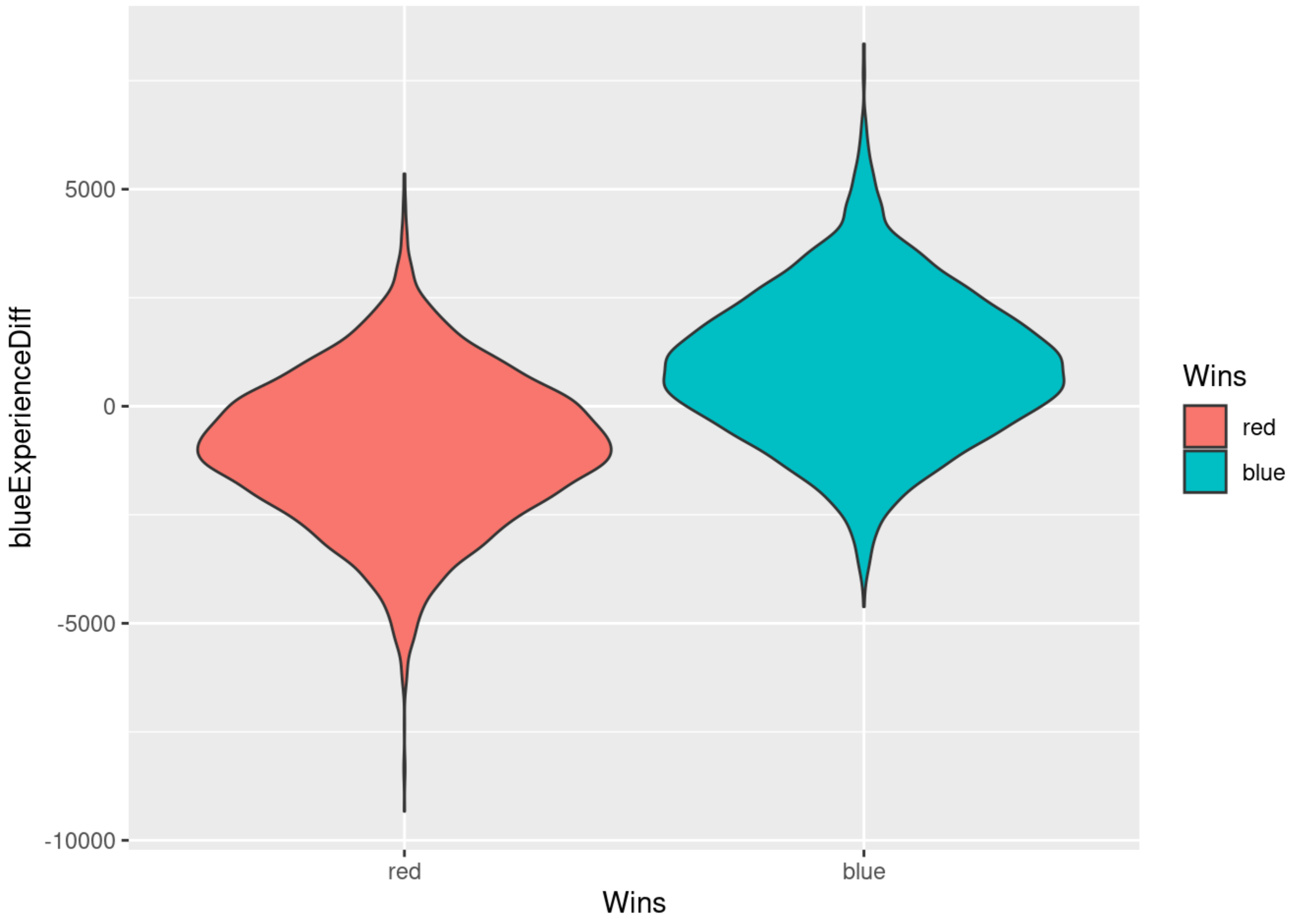
```
## [1] 2.009182e-38
```

```
Very small number. We can reject the null hypothesis. We can conclude that getting bo
th dragon and herald can definitly increase the winrate.
```

Experience diff

```
The experience difference a team has is positively related to the their performance i
n game. If a team has a positive experience diff in comparison to the other team (cur
rent - opposing), they exhibit greater frequency of wins.
```

```
ggplot(data = data, aes(x = Wins, y = blueExperienceDiff, fill = Wins)) + geom_violin
()
```



```
t.test(dataBlueWins$blueTotalExperience, dataBlueWins$redTotalExperience)
```

```
##
##  Welch Two Sample t-test
##
## data:  dataBlueWins$blueTotalExperience and dataBlueWins$redTotalExperience
## t = 40.784, df = 9829.1, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  864.6201 951.9288
## sample estimates:
## mean of x mean of y
##  18404.58  17496.30
```

```
t.test(dataRedWins$blueTotalExperience, dataRedWins$redTotalExperience)
```

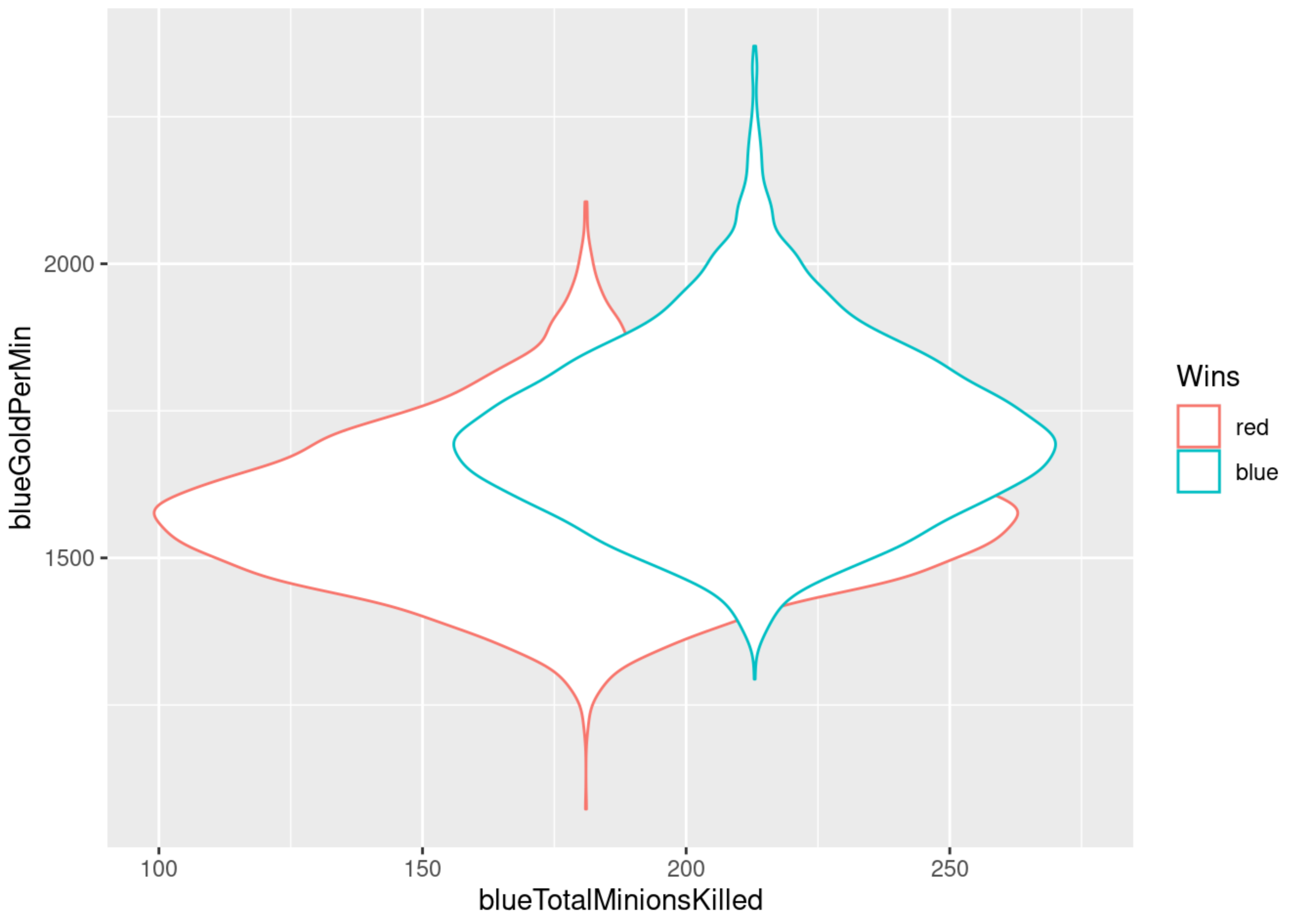
```
##
##  Welch Two Sample t-test
##
## data:  dataRedWins$blueTotalExperience and dataRedWins$redTotalExperience
## t = -43.886, df = 9871.6, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1015.3092  -928.4887
## sample estimates:
## mean of x mean of y
##  17453.47  18425.37
```

Gold diff

```
The creep score of a team vs its gold gain rate per minute are positively related. Th
e more minions a team kills, the quicker and more gold their tend to gain.
```

```
ggplot(data = data, aes(x = blueTotalMinionsKilled, y = blueGoldPerMin, color = Win
s)) + geom_violin()
```

```
## Warning: position_dodge requires non-overlapping x intervals
```



Wards Placed

```
meanPerGame <- function(default = 1){

  return( (data$blueWardsPlaced[default] + data$redWardsPlaced[default]) )

}
means <- sapply(1:nrow(data), meanPerGame)

mean(means)
```

```
## [1] 44.65624
```

```
max(dataBlueWins$blueWardsPlaced)
```

```
## [1] 250
```

```
max(dataBlueWins$redWardsPlaced)
```

```
## [1] 268
```

```
max(dataRedWins$redWardsPlaced)
```

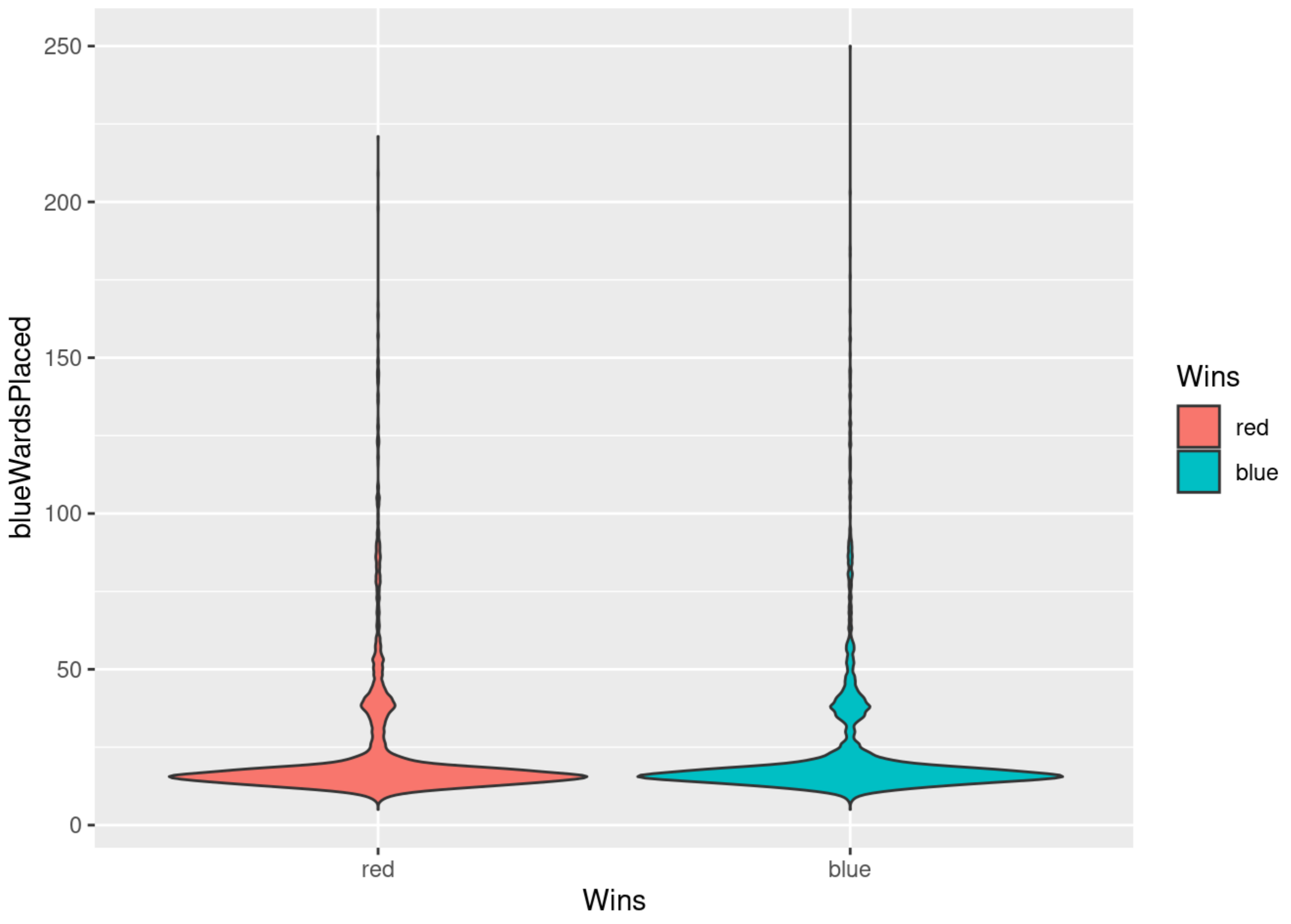
```
## [1] 276
```

```
max(dataRedWins$blueWardsPlaced)
```

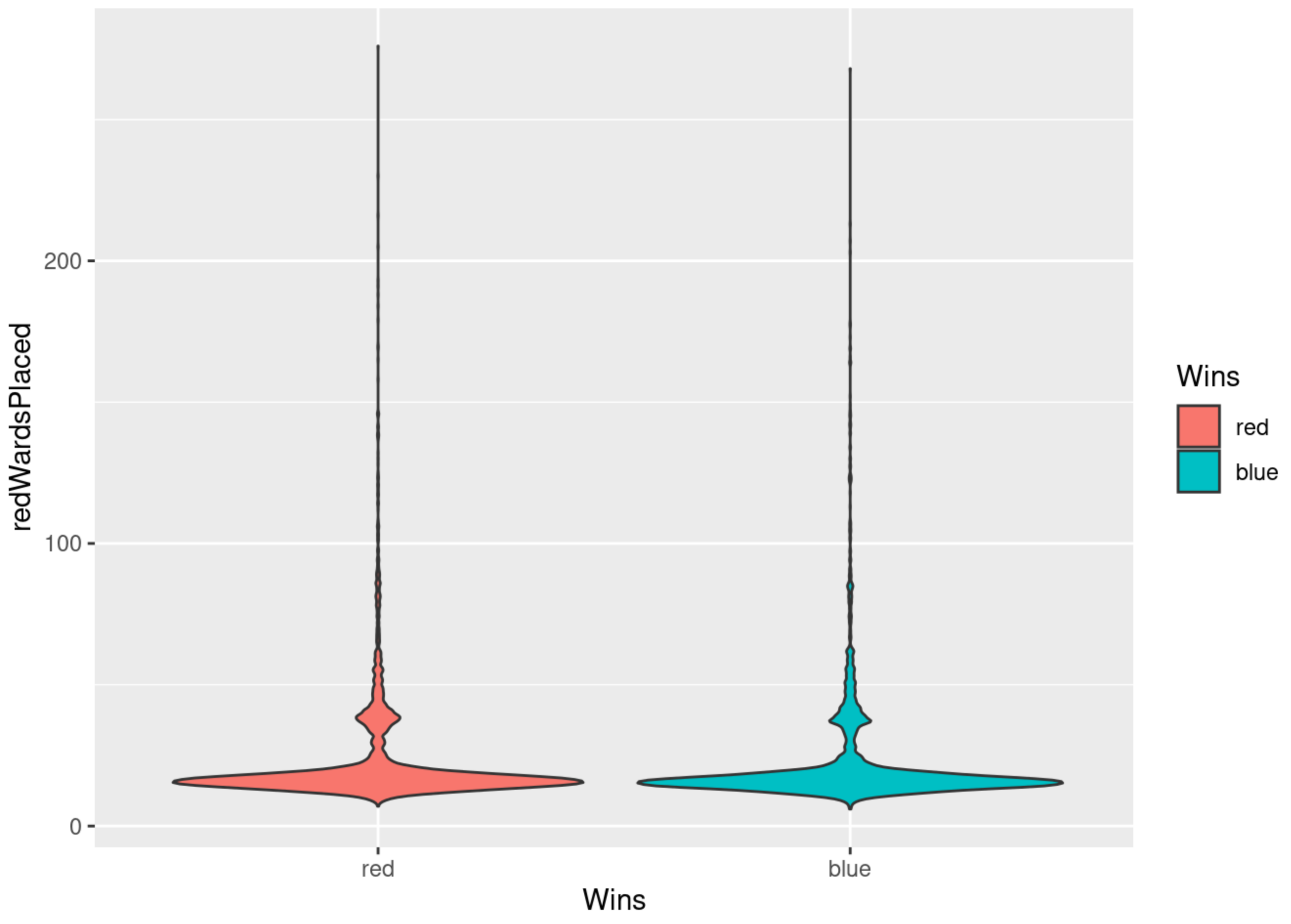
```
## [1] 221
```

```
Plots of the Distribution of Wards based upon wins-- despite winning or losing, neith
er team shows any inherent advantage based upon their wards.
```

```
ggplot(data = data, aes(x = Wins, y = blueWardsPlaced, fill = Wins)) + geom_violin()
```



```
ggplot(data = data, aes(x = Wins, y = redWardsPlaced, fill = Wins)) + geom_violin()
```



```
length(which(data$Wins == "blue"))
```

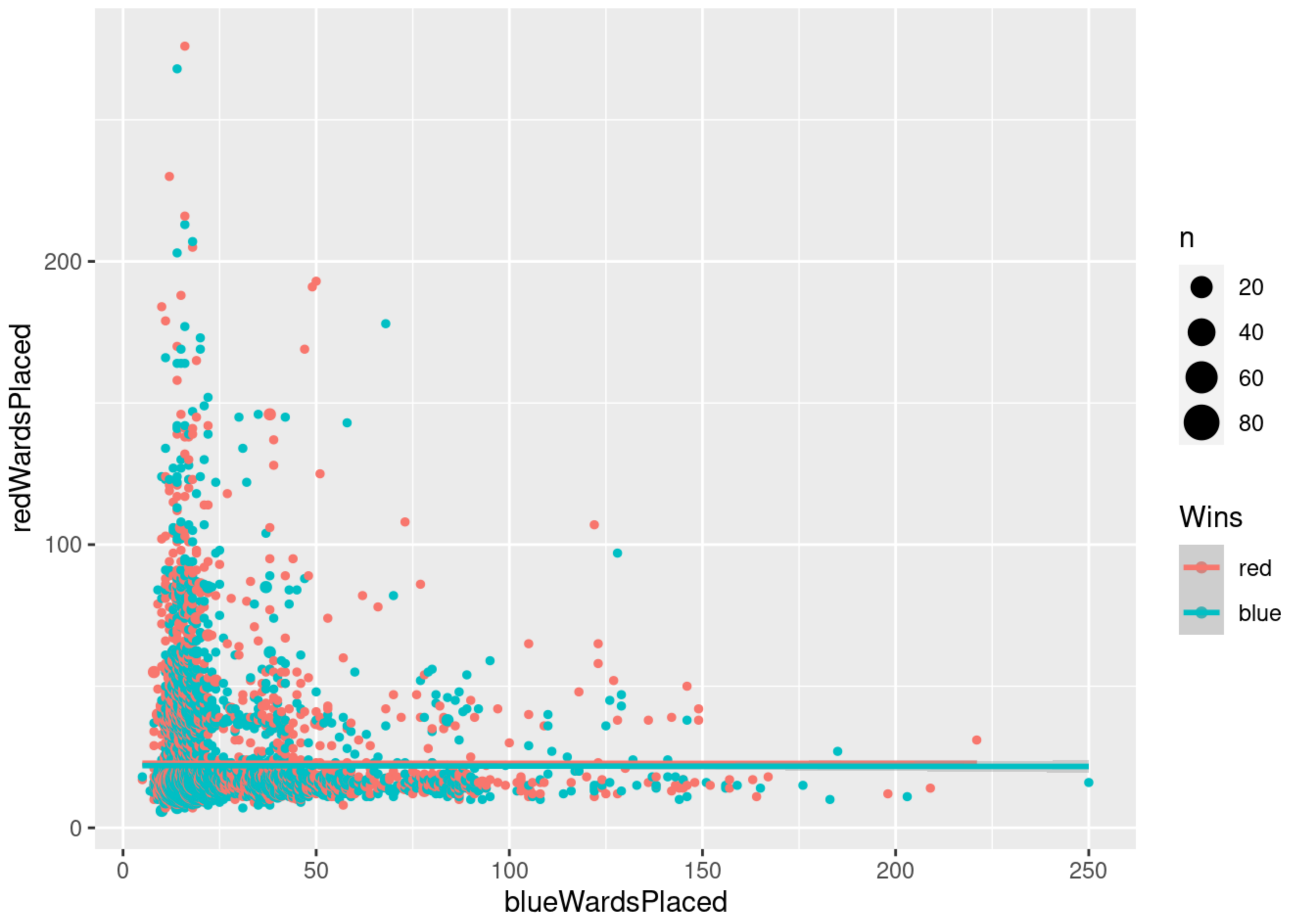
```
## [1] 4930
```

```
length(which(data$Wins == "red"))
```

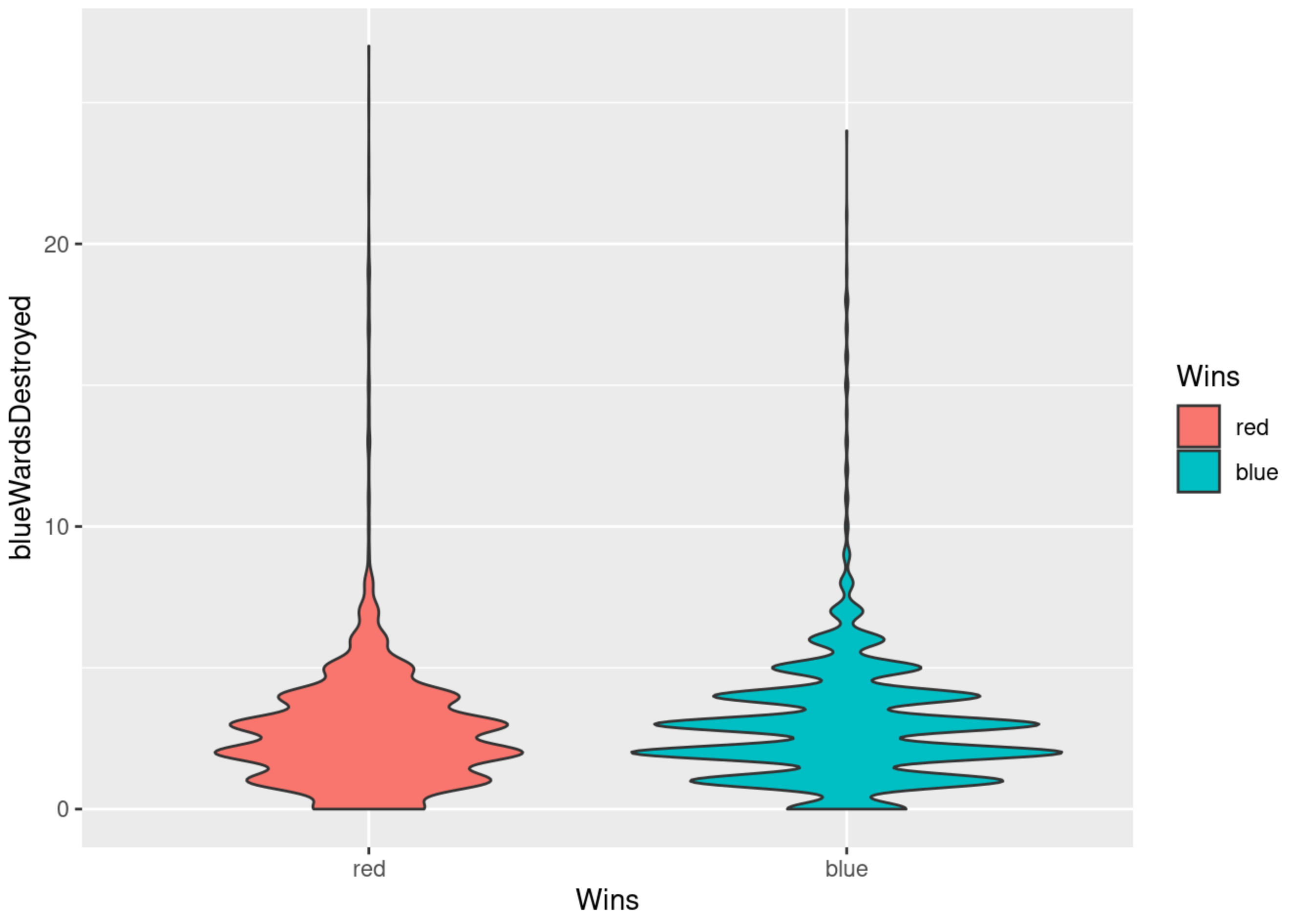
```
## [1] 4949
```

```
ggplot(data = data, aes(blueWardsPlaced, redWardsPlaced, color = Wins)) + geom_count
() + geom_smooth()
```

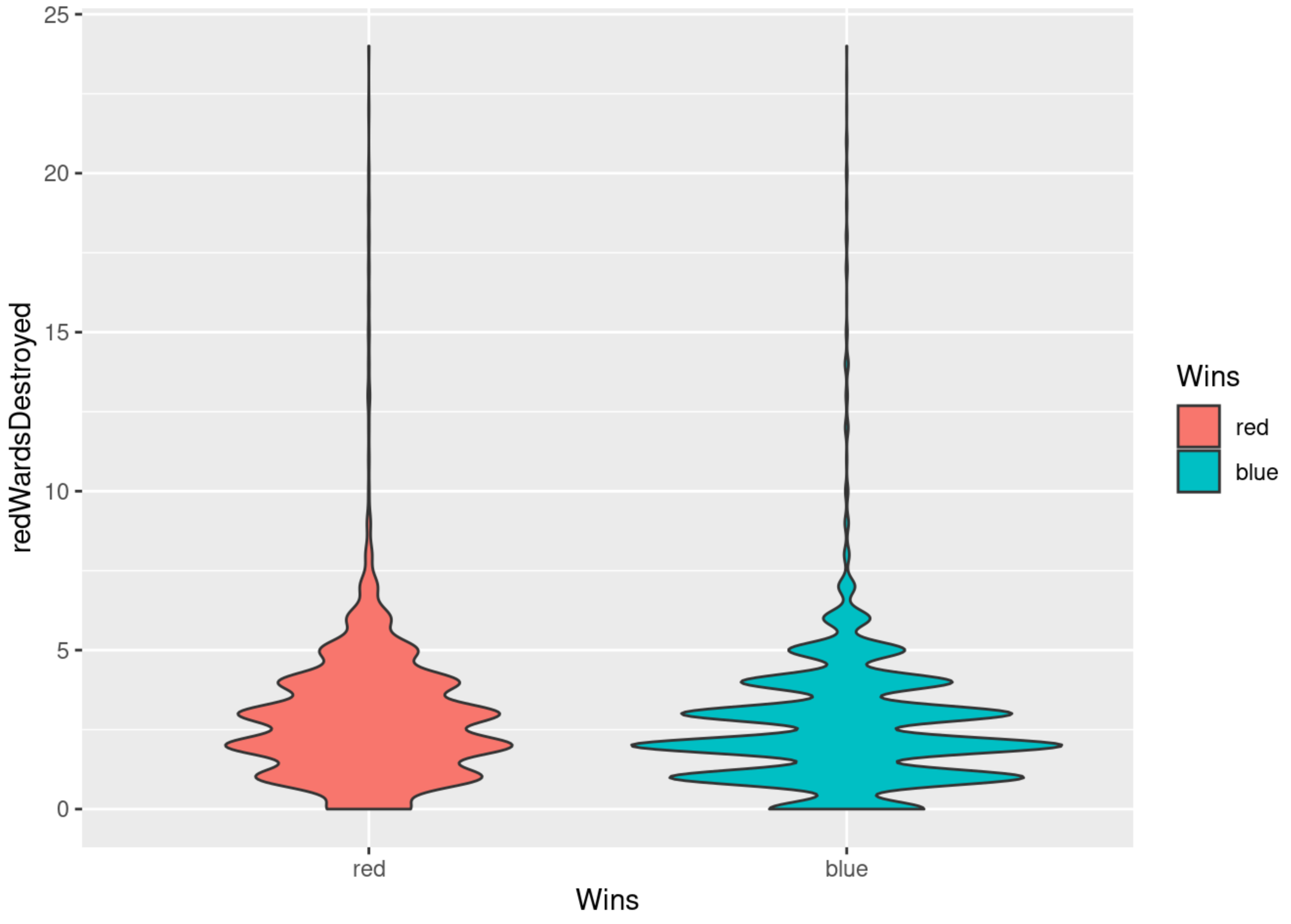
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
ggplot(data = data, aes(x = Wins, y = blueWardsDestroyed, fill = Wins)) + geom_violin
()
```

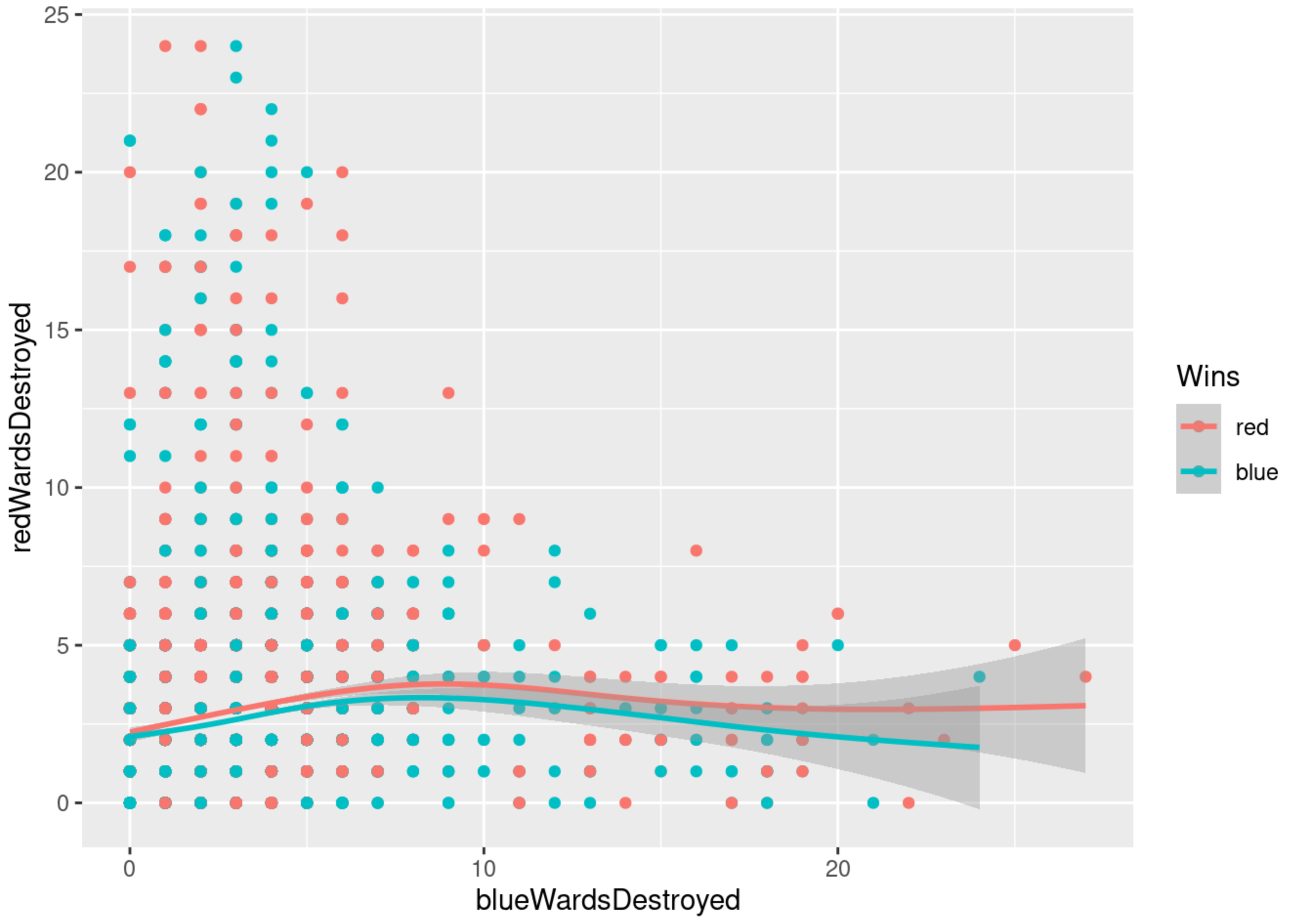


```
ggplot(data = data, aes(x = Wins, y = redWardsDestroyed, fill = Wins)) + geom_violin
()
```



```
ggplot(data = data, aes(blueWardsDestroyed, redWardsDestroyed, color = Wins)) + geom_
point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
Boostrap assumptions & t Test for the mean of blue wins, all three seem to show littl
e correlation between performance improvement and placing & destroying wards.
```

```
x <- function(default=1){
  resample <- sample(dataBlueWins$blueWardsPlaced, length(dataBlueWins), replace = T)
  return(mean(resample))
}
functionData <- sapply(1:10000, x)

sum(functionData > mean(dataRedWins$blueWardsPlaced)) / length(data)
```

```
## [1] 113.2
```

```
y <- function(default=1){
  resample <- sample(dataBlueWins$blueWardsDestroyed, length(dataBlueWins), replace =
T)
  return(mean(resample))
}
functionData <- sapply(1:10000, y)
sum(functionData > mean(dataRedWins$blueWardsDestroyed)) / length(data)
```

```
## [1] 173.125
```

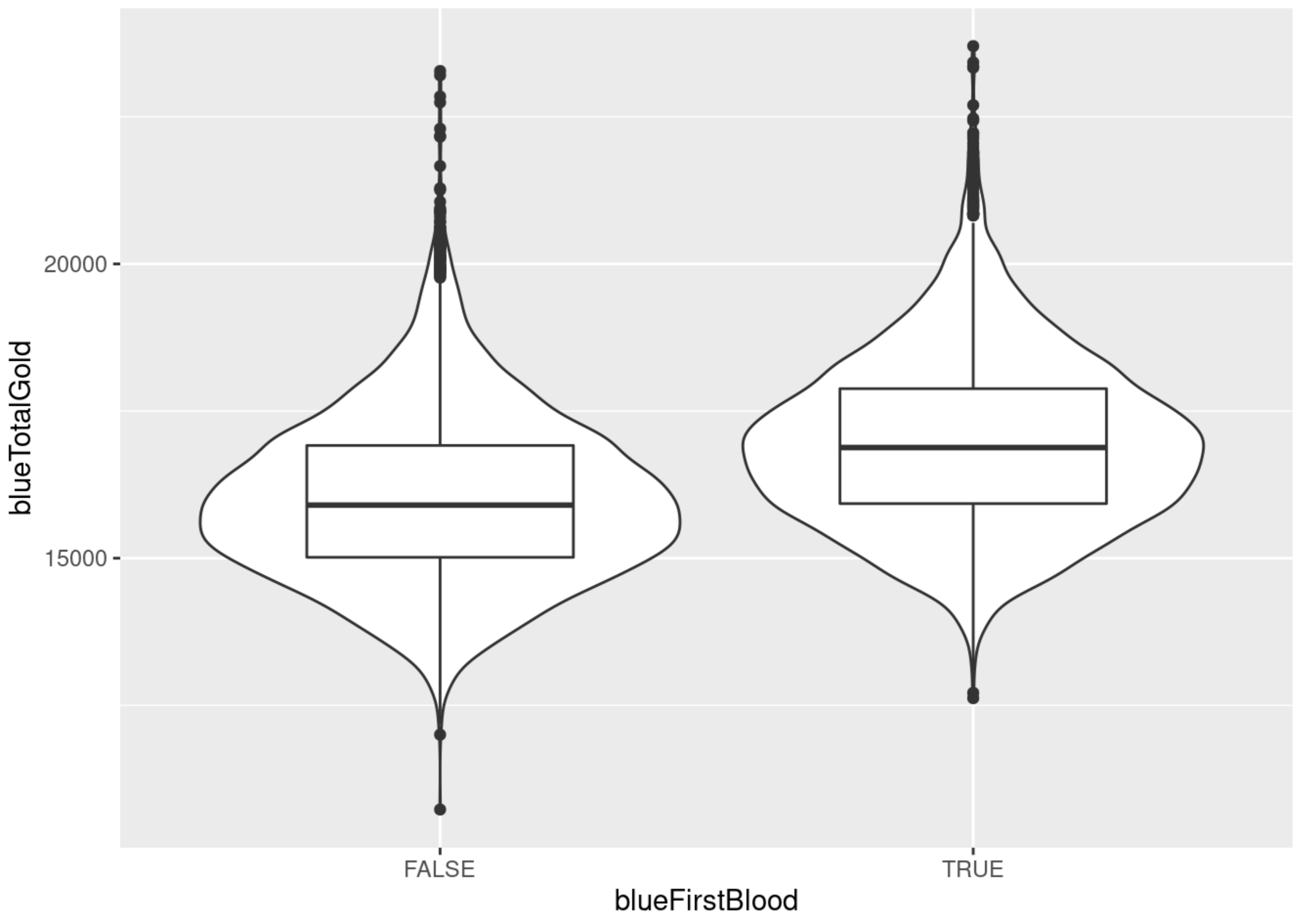
```
t.test(dataBlueWins$blueWardsPlaced,dataRedWins$blueWardsPlaced)$p.value
```

```
## [1] 0.9931048
```

Kills & First Blood

```
Based upon these violin / box plots, we can assume an inverse relationship between re
d deaths and red performance in game. The more deaths red has, the less their frequen
cy of winning.
```

```
ggplot(data = data, aes(x = blueFirstBlood, y = blueTotalGold)) + geom_violin() + geo
m_boxplot(width = 0.5)
```

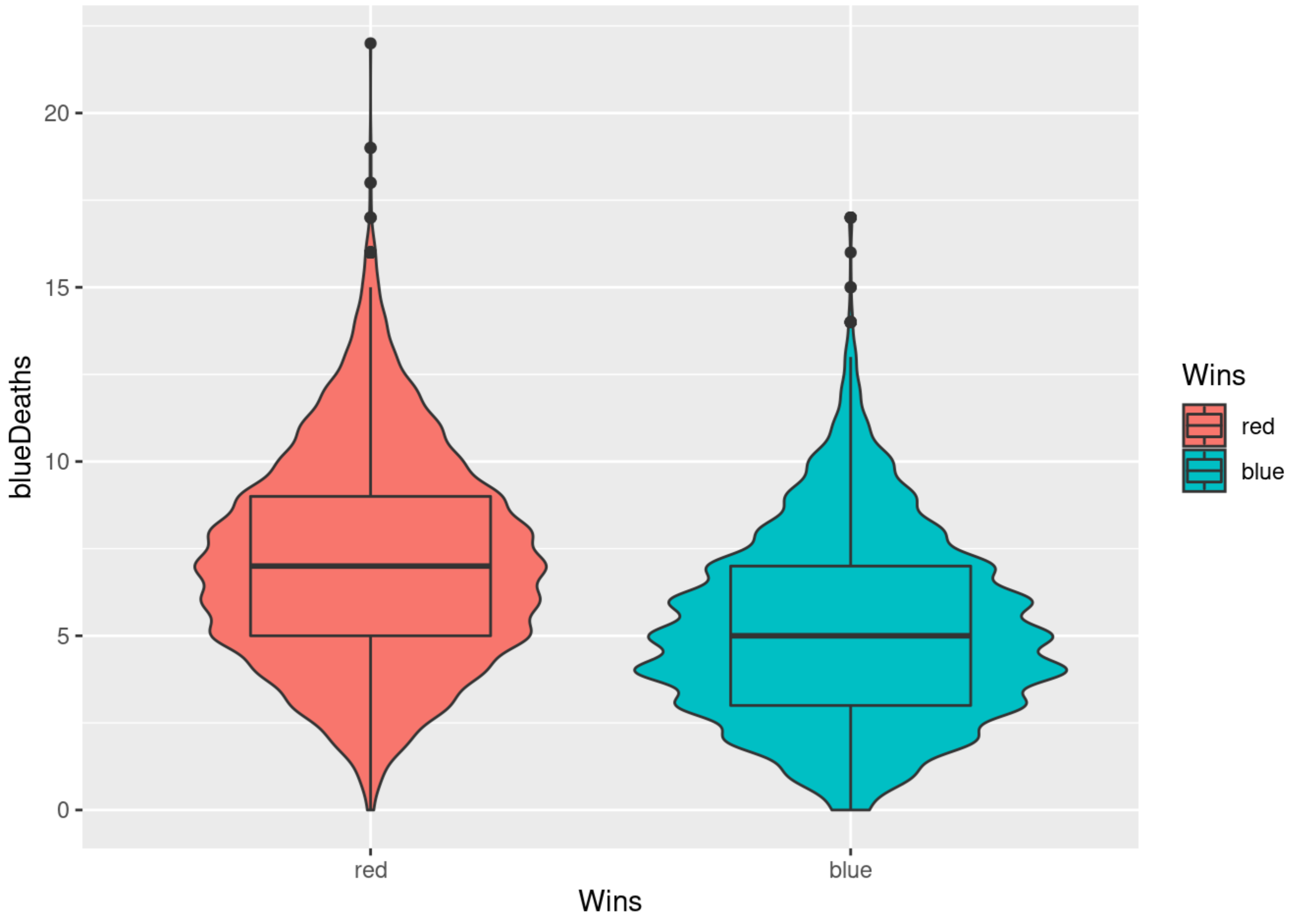


```
t.test(data$blueTotalGold, as.numeric(data$blueFirstBlood))$p.value
```

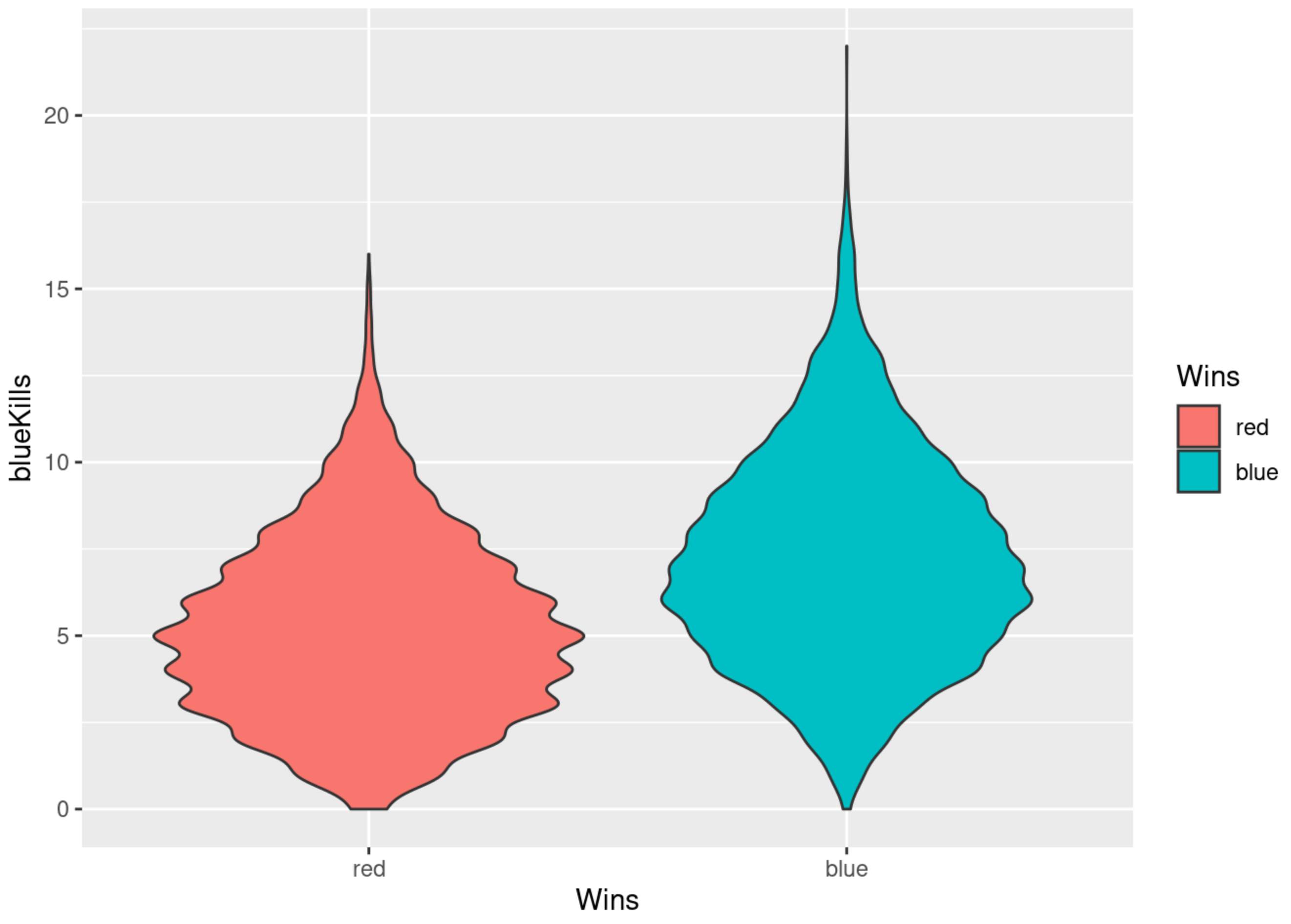
```
## [1] 0
```

```
Deaths appear to have a more significant effect on the outcome of the game. The same
assumption applies to blue.
```

```
ggplot(data = data, aes(x = Wins, y = blueDeaths, fill = Wins)) + geom_violin() + geo
m_boxplot(width = 0.5)
```



```
ggplot(data = data, aes(x = Wins, y = blueKills, fill = Wins)) + geom_violin()
```



Models and Analyze

Setup Train and Test

```
set.seed(12345)
data$ID <- 1:nrow(data)
train <- sample_frac(data,0.6)
test <- anti_join(data,train,by="ID")
train$ID <- NULL
test$ID <- NULL
```

k Nearest Neighbors

```
knnTrain <- train
knnTest <- test
library(class)
train_Wins <- knnTrain$Wins
test_Wins <- knnTest$Wins
knnTrain$Wins <- NULL
knnTest$Wins <- NULL
knnTrain$blueFirstBlood <- NULL
knnTest$blueFirstBlood <- NULL
knnTrain$redFirstBlood <- NULL
knnTest$redFirstBlood <- NULL
```

```
predicted_Wins <-knn(knnTrain,knnTest,train_Wins,k=1,prob=TRUE)

table(predicted_Wins,test_Wins)
```

```
##               test_Wins
## predicted_Wins  red blue
##           red  1066  928
##           blue  879 1079
```

K nearest neighbors seems to get the result a little over half right. Unfortunately, if we increase k, the model only gets less accurate.

Logistic Regression

Logistic Regression does not particularly help us. There isn’t a significant division between any groups from the observations in the data set, so we omitted the model from use.

Decision Trees

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

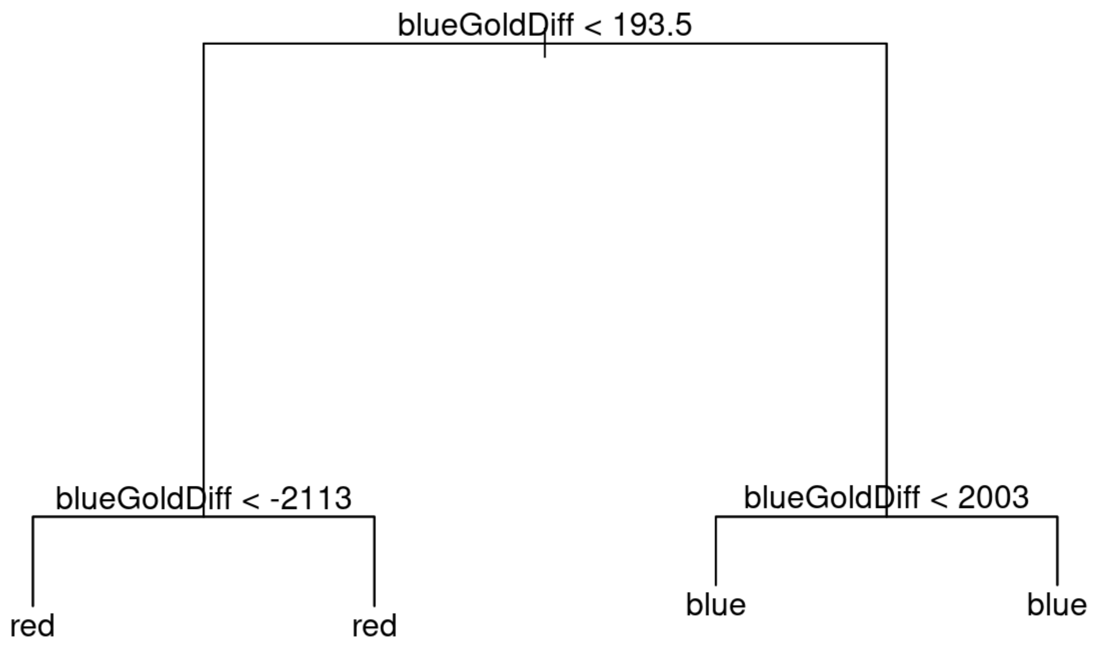
```
model <- tree(Wins ~ ., data = train)
prune.misclass(model, best = 2)
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 5927 8215 red ( 0.5068 0.4932 )
##   2) blueGoldDiff < 193.5 3143 3731 red ( 0.7194 0.2806 ) *
##   3) blueGoldDiff > 193.5 2784 3230 blue ( 0.2669 0.7331 ) *
```

```
summary(model)
```

```
##
## Classification tree:
## tree(formula = Wins ~ ., data = train)
## Variables actually used in tree construction:
## [1] "blueGoldDiff"
## Number of terminal nodes:  4
## Residual mean deviance:  1.105 = 6543 / 5923
## Misclassification error rate: 0.2742 = 1625 / 5927
```

```
plot(model)
text(model, pretty = 0)
```



```
predictedWins <- predict(model, test, type ="class")
actualWins <- test$Wins

table(predictedWins, actualWins)
```

```
##              actualWins
## predictedWins  red blue
##          red  1480  624
##          blue  465 1383
```

This model gives us about a 73% accuracy, which is interesting considering exactly one decision is made on this decision tree. The gold difference between teams would have a large effect on games, since gold is used to buy items which can give players an advantage, and the larger the gold difference is, the bigger the advantage of the team which is ahead.

Random Forests

```
library(randomForest)
```

```
## randomForest 4.6-14
```

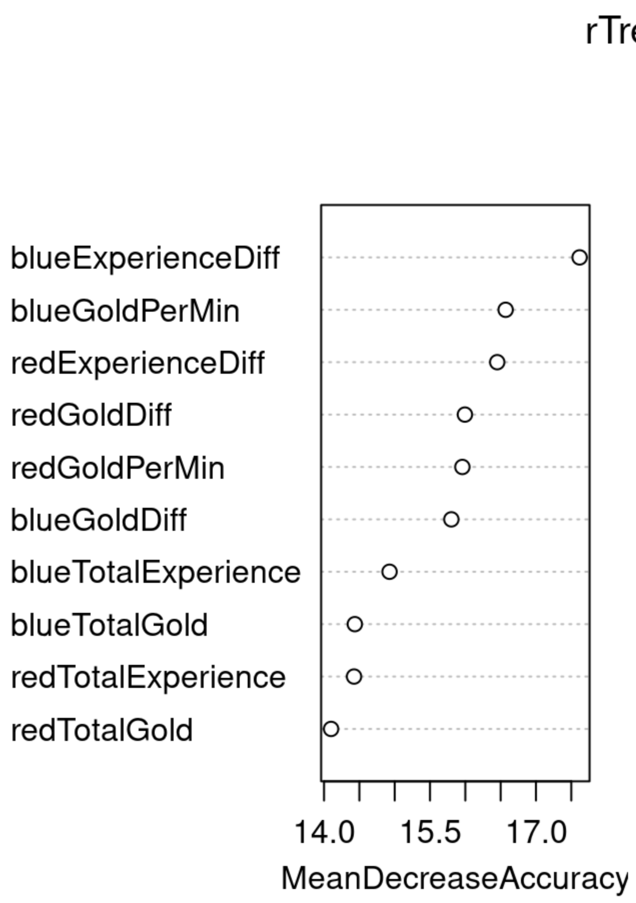
```
## Type rfNews() to see new features/changes/bug fixes.
```

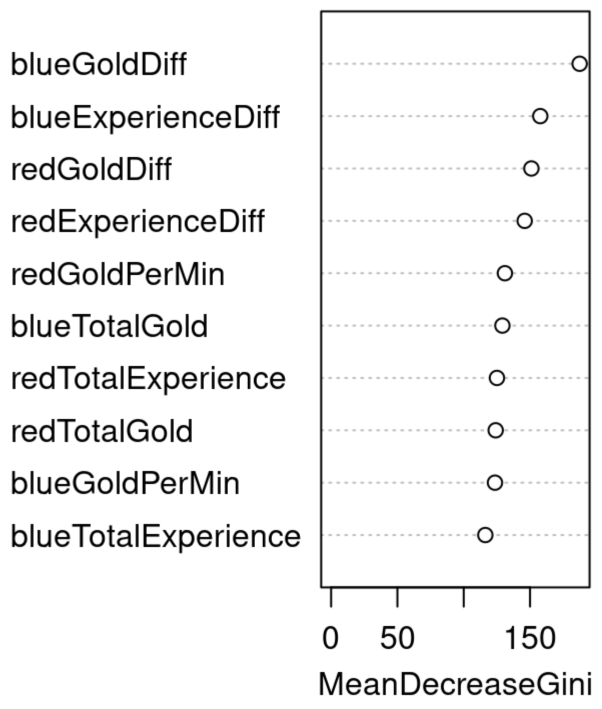
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
train$ID <- NULL
test$ID <- NULL
rTree <- randomForest(Wins ~ ., data = train, ntree = 200, mtry = 2, importance = TRU
E)
varImpPlot(rTree, n.var= 10, sort = T, scale = T)
```





```
predictedWins <- predict(rTree, test)
actualWins <- test$Wins


table(predictedWins, actualWins)
```

```
##              actualWins
## predictedWins  red blue
##          red  1417  568
##          blue  528 1439
```

```
Despite this data originating from the first 10 minutes of gameplay in high-tier game
s, the random forest does extremely well in its accuracy. Holding approximately a 75%
accuracy rating, the random forest model does well in predicting the team which ultim
ately wins the game. We used this model because it typically tends to perform at the
very least on par with if not much better than the other models we typically use, and
not only this, we get a much more in-depth look at the factors which the model relies
most on. For instance, the gold and experience differences between the two teams seem
ed to contribute most to the prediction, which makes sense in game. The team with the
most gold can afford to buy the required equipment for their champions faster and the
n outperform the opposing team, which leads to an increase in the rate of experience.
```

Conclusions & Further Questions

From our analysis, we can conclude that gold and experience are the two largest factors in determining the winner of a game. Since gold and experience are directly related to what goes on in a game and largely related to other variables that we analyze here, we tried running a version of our models with all of the gold and experience information set to NULL, and to our surprise, the models actually significantly decreased in accuracy. This indicates that experience and gold are, to some extent, independent from the other variables in this dataset. This is most likely due to the fact that there are ways to gain experience and gold that are not directly related to other variables, such as through the use of items that boost experience when used or double the amount of gold going to a team. Unfortunately, this dataset does not give us any information about what items the team bought. An analysis on a dataset which includes that information might indicate whether the type and stats of an item changes the chances of a given team winning.

The random forest model gave us the most accurate prediction, with the decision tree close behind. K nearest numbers came in a distant third with a little over 50% accuracy, and the linear regression model did not work at all for this set of data.

These models would be interesting and useful to gamers who play League of Legends and teams hoping to improve their strategy. Players would be able to focus on a strategy that increases their experience and gold, therefore increasing their chances of winnning.