Exploratory Data Analysis of League of Legends Player Statistics and Game Outcome

The first step is to clean the data. The data comes in wide format, it was from web scraping. Need to deal with special chars from the scraping. Also replacing missing values with averages.

```
library(tibble)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.2
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.0     v stringr   1.5.1
## v lubridate 1.9.3     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
#read in csv
df <-read.csv("diamond_preclean.csv")


#if r complains about \0use linux command to clean file: sed -i 's/\x0//g' file.csv, this only happened

#turn dataframe into tibble
league_data <- as_tibble(df)
head(league_data)
```

```
## # A tibble: 6 x 91
##   Win.Loss Player1.name   Player1.champ.name Player1.overall.wr
##   <chr>    <chr>          <chr>              <chr>
## 1 loss     Kaizer Morde   viego              50
## 2 win      Elo Demon      ornn               58
## 3 win      PogoDuMaxi     warwick            53
## 4 loss     comedivetoplv3 fiora              53
## 5 loss     TAG Caynn      viego              None
## 6 loss     BBearDude      volibear           51
## # i 87 more variables: Player1.total.games <chr>, Player1.champ.games <chr>,
## #   Player1.champ.wr <chr>, Player1.kda <chr>, Player1.cs <dbl>,
## #   Player1.mastery <chr>, Player2.name <chr>, Player2.champ.name <chr>,
## #   Player2.overall.wr <chr>, Player2.total.games <chr>,
## #   Player2.champ.games <chr>, Player2.champ.wr <chr>, Player2.kda <chr>,
## #   Player2.cs <dbl>, Player2.mastery <chr>, Player3.name <chr>,
## #   Player3.champ.name <chr>, Player3.overall.wr <chr>, ...
```

```r
#select kda columns frome every player
kda_columns <-select(league_data, ends_with("kda"))

#kda came with two dots, second one needs to be removed
rm_dot <- function(x){
  str_replace(x, "^([0-9]+\\.[0-9]+)\\.", "\\1")
}

#call remove dot function on all columns
kda_columns <- kda_columns %>%
    mutate_all(rm_dot)


#remove kda columns in original dataset
league_data <- league_data %>%
  select(-ends_with("kda"))

#add back in columns with correct num of .
league_data <- bind_cols(league_data, kda_columns)



#create function to calculate mean of columns
column_mean <- function(x)
{
  nums <- suppressWarnings(as.numeric(as.character(x))) #supressing warnings because as.numeric is inse
  m <- mean(nums, na.rm = T)
  return(m)
}

#league_data

#call function on all kda columns
kda.col.results<- league_data %>%
  summarise(across(ends_with("kda"), column_mean))

overall.wr.col.results <-  league_data %>%
  summarise(across(ends_with("overall.wr"), column_mean))

total.games.col.results <-  league_data %>%
  summarise(across(ends_with("total.games"), column_mean))

champ.games.col.results <-  league_data %>%
  summarise(across(ends_with("champ.games"), column_mean))

champ.wr.col.results <-  league_data %>%
  summarise(across(ends_with("champ.wr"), column_mean))

cs.col.results <-  league_data %>%
  summarise(across(ends_with(".cs"), column_mean))

mastery.col.results <-  league_data %>%
  summarise(across(ends_with(".mastery"), column_mean))
```

```r
kda.mean <-rowMeans(kda.col.results)
total.games.mean <-rowMeans(total.games.col.results)
overall.wr.mean <- rowMeans(overall.wr.col.results)
champ.games.mean <- rowMeans(champ.games.col.results)
champ.wr.mean <- rowMeans(champ.wr.col.results)
cs.mean <- rowMeans(cs.col.results)
mastery.mean <- rowMeans(mastery.col.results)

#standardize data
league_data <- mutate(league_data, across(ends_with("kda"), ~str_replace(., "Perfect", ""))) %>%
  mutate(across(ends_with("overall.wr"), ~str_replace(., "None_unranked", "None"))) %>%
  mutate(across(ends_with("total.games"), ~str_replace(., "None_unranked", "0"))) %>%
  mutate(across(ends_with("mastery"), ~str_replace(., "n/a", "0")))  %>%#when no mastery they have 0
  mutate(across(ends_with("mastery"), ~str_replace(., "ng", "0")))



#keep only unique rows incase there are duplicates. Should only be a few duplicates max anyway
league_data <- league_data %>% distinct()

#store tibble pre adding in averages for visualization
raw_data <- league_data

#now filling in averages if they dont have games on that champ
league_data <-
  mutate(league_data, across(ends_with("kda"), ~str_replace(., "None", as.character(sprintf("%.2f", kda
  mutate(across(ends_with("overall.wr"), ~str_replace(., "None", as.character(sprintf("%.2f", overall.wr
  mutate(across(ends_with("total.games"), ~str_replace(., "None", as.character(sprintf("%.2f", total.gam
  mutate(across(ends_with("champ.games"), ~str_replace(., "None", "0"))) %>% #makes more sense for this
  mutate(across(ends_with("champ.wr"), ~str_replace(., "None", as.character(sprintf("%.2f", champ.wr.mea
  mutate(across(ends_with("cs"), ~str_replace(., "None", as.character(sprintf("%.2f", cs.mean)))))


head(league_data)
```

```
## # A tibble: 6 x 91
##   Win.Loss Player1.name   Player1.champ.name Player1.overall.wr
##   <chr>    <chr>          <chr>              <chr>
## 1 loss     Kaizer Morde   viego              50
## 2 win      Elo Demon      ornn               58
## 3 win      PogoDuMaxi     warwick            53
## 4 loss     comedivetoplv3 fiora              53
## 5 loss     TAG Caynn      viego              52.77
## 6 loss     BBearDude      volibear           51
## # i 87 more variables: Player1.total.games <chr>, Player1.champ.games <chr>,
## #   Player1.champ.wr <chr>, Player1.cs <chr>, Player1.mastery <chr>,
## #   Player2.name <chr>, Player2.champ.name <chr>, Player2.overall.wr <chr>,
## #   Player2.total.games <chr>, Player2.champ.games <chr>,
## #   Player2.champ.wr <chr>, Player2.cs <chr>, Player2.mastery <chr>,
## #   Player3.name <chr>, Player3.champ.name <chr>, Player3.overall.wr <chr>,
## #   Player3.total.games <chr>, Player3.champ.games <chr>, ...
```

```
write_csv(league_data, "example.csv")

sprintf("%f is overall winrate mean",overall.wr.mean)
```

```
## [1] "52.768905 is overall winrate mean"
```

```
sprintf("%f is champ winrate mean",champ.wr.mean)
```

```
## [1] "54.010401 is champ winrate mean"
```

```
sprintf("%f is total games mean",total.games.mean)
```

```
## [1] "265.881343 is total games mean"
```

```
sprintf("%f is champ games mean",champ.games.mean)
```

```
## [1] "66.529117 is champ games mean"
```

```
sprintf("%f is champ kda mean",kda.mean)
```

```
## [1] "2.804662 is champ kda mean"
```

```
sprintf("%f is cs winrate mean",cs.mean)
```

```
## [1] "150.545245 is cs winrate mean"
```

Fixing format

- to make rows usable for visualizations, to achieve this pivot longer
- turning all the columns to numeric
- making sure there are only unique rows

```
raw_data <- pivot_longer(raw_data, cols = -Win.Loss, names_to = c("Player", ".value"),
                         names_pattern = "Player([0-9]+).(.*)")

#remove rows that have none in it for visualization
league_long <- raw_data[raw_data$kda!="None", ]


#the case of overall winrate being none can happen even if other info wasn't found, so remove those row
league_long <- league_long[league_long$overall.wr!="None", ]

league_long$overall.wr <- as.integer(league_long$overall.wr)
league_long$total.games <- as.integer(league_long$total.games)
league_long$champ.games <- as.integer(league_long$champ.games)
league_long$champ.wr <- as.integer(league_long$champ.wr)
league_long$kda <- as.numeric(league_long$kda)
head(league_long)
```

```
## # A tibble: 6 x 11
##   Win.Loss Player name    champ.name overall.wr total.games champ.games champ.wr
##   <chr>    <chr>  <chr>   <chr>           <int>       <int>       <int>    <int>
## 1 loss     1      Kaizer~ viego              50         433         247       47
## 2 loss     2      Strawb~ kindred            52         572         226       52
## 3 loss     3      Veiga   veigar             52         405         386       53
## 4 loss     4      BIE CH~ xayah              58         153           3        0
## 5 loss     5      Dal By~ alistar            53         617         100       57
## 6 loss     6      Lizzy4~ renekton           51         462          66       62
```

```
## # i 3 more variables: cs <dbl>, mastery <chr>, kda <dbl>
```

```
league_long <- unique(league_long)
```

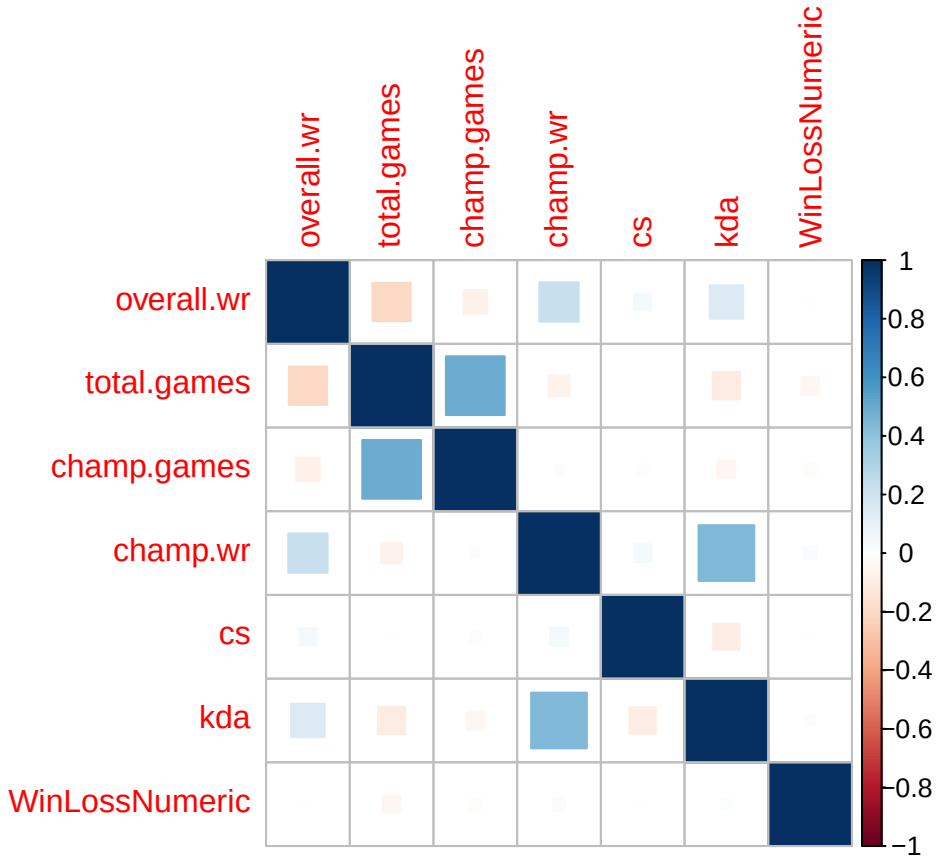
Creating Correlation matrix

- Encoding win loss column to numeric 1 or 0
- creating correlation matrix to see variables that are correlated with each other and see if indivudla player statistics before combining into team statistics are correlated with target var

```
league_long_corr <- league_long
league_long_corr$WinLossNumeric <- ifelse(league_long$Win.Loss == "win", 1, 0)

# Correlation matrix for numerical features
numeric_cols <- select(league_long_corr, where(is.numeric))
cor_matrix <- cor(numeric_cols, use = "complete.obs")
cor_matrix
```

```
##                  overall.wr  total.games champ.games    champ.wr           cs
## overall.wr      1.000000000 -0.209725808 -0.07991200  0.22093539  0.040662695
## total.games    -0.209725808  1.000000000  0.49249252 -0.06269494 -0.002543192
## champ.games    -0.079911995  0.492492516  1.00000000  0.01123034  0.016599857
## champ.wr        0.220935389 -0.062694940  0.01123034  1.00000000  0.046009448
## cs              0.040662695 -0.002543192  0.01659986  0.04600945  1.000000000
## kda             0.159607743 -0.106979277 -0.04378285  0.44293833 -0.098950712
## WinLossNumeric -0.008824522 -0.042403027 -0.01995784  0.02218678  0.005594534
##                         kda WinLossNumeric
## overall.wr      0.15960774   -0.008824522
## total.games    -0.10697928   -0.042403027
## champ.games    -0.04378285   -0.019957843
## champ.wr        0.44293833    0.022186777
## cs             -0.09895071    0.005594534
## kda             1.00000000    0.014968343
## WinLossNumeric  0.01496834    1.000000000
```

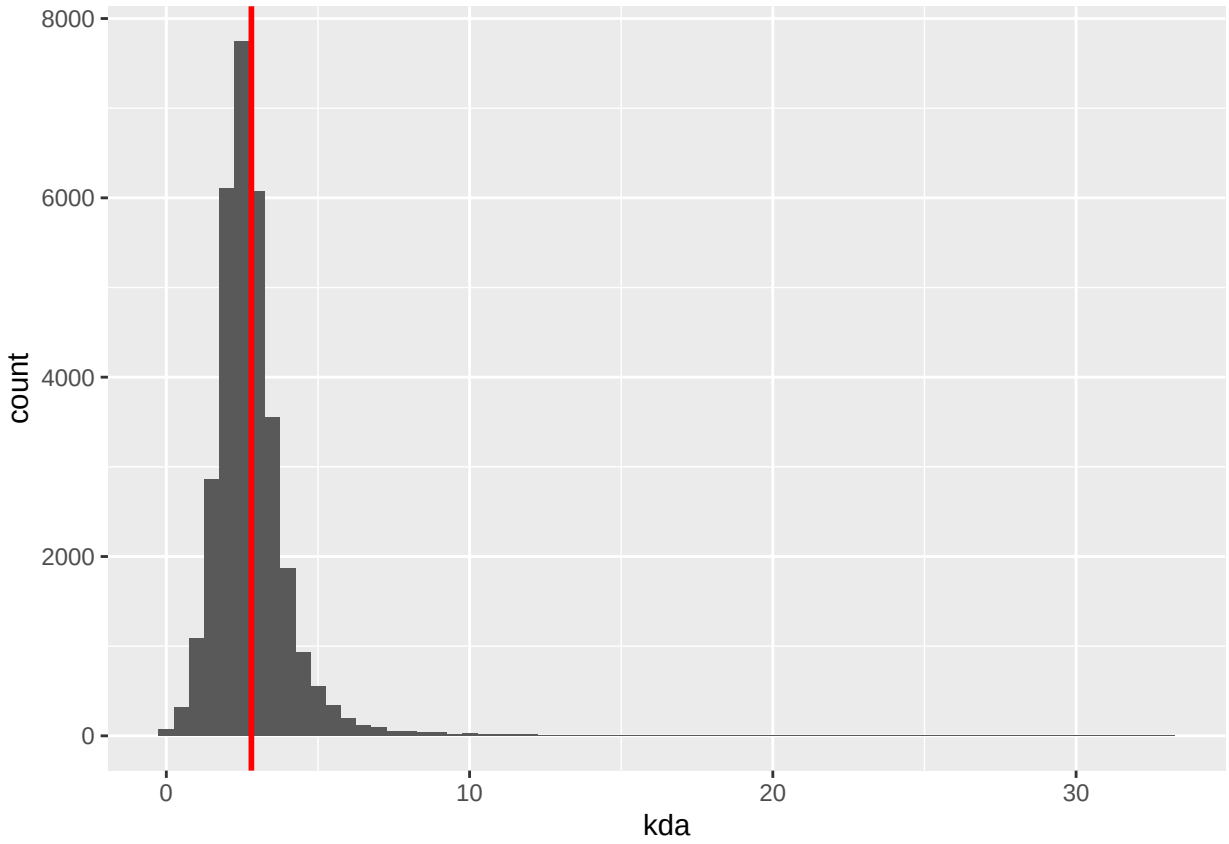
```
corrplot::corrplot(cor_matrix, method = "square")
```



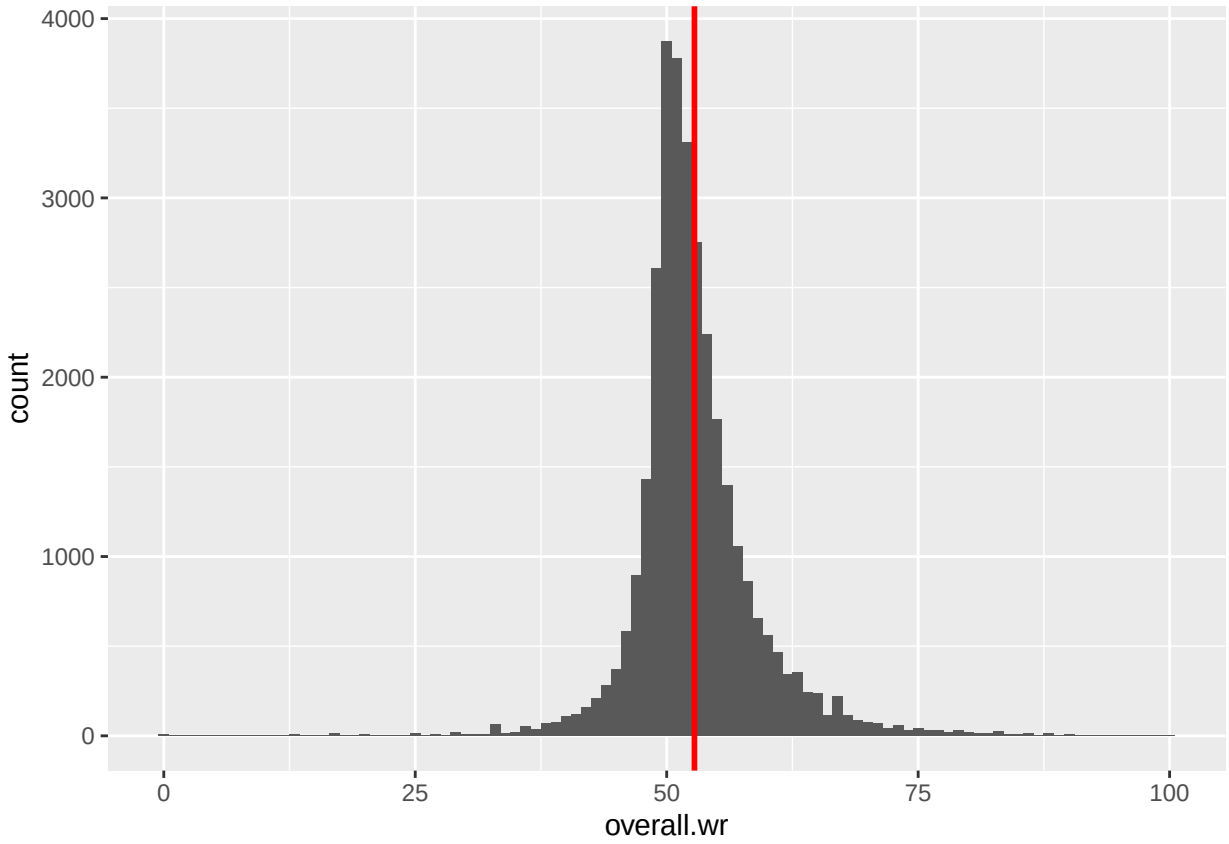
Creating visuals of dependent variables

- Creating histograms with a line to show average value
- Creating density plot of champ winrates
- Boxplots are good information to see split between win and loss for values of independent var

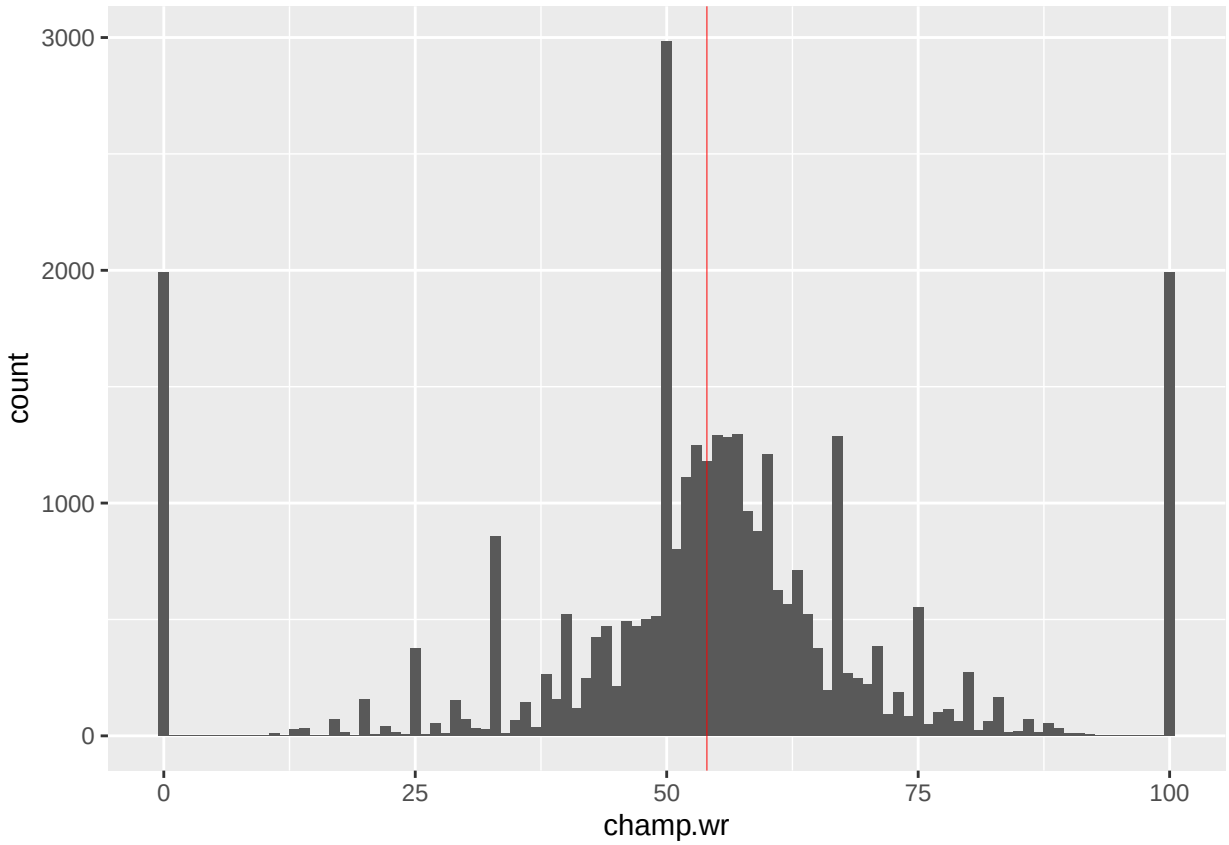
```
#creation of histograms
ggplot(league_long, aes(x = kda)) +
   geom_histogram(binwidth = .5)  +
   geom_vline(aes(xintercept = mean(kda)),
              color = "red", linewidth = 1)
```



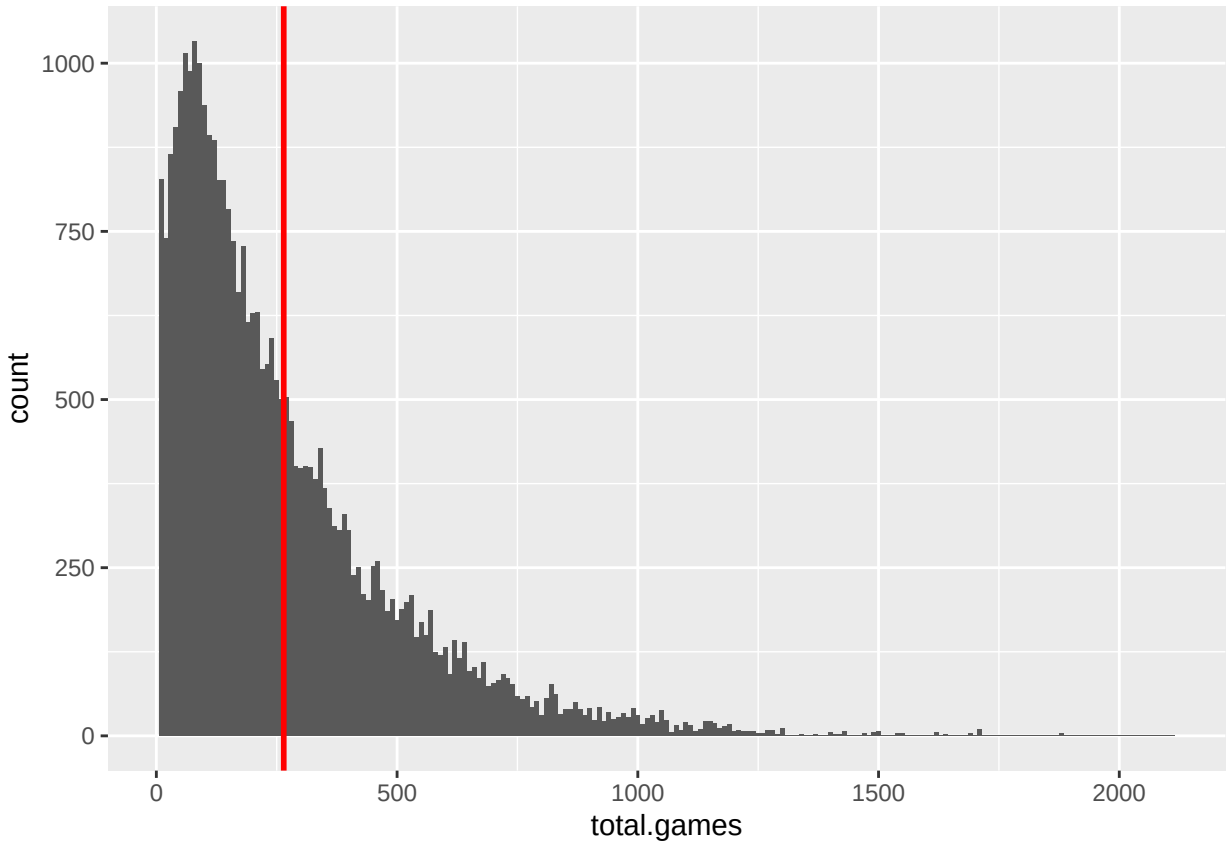

```
ggplot(league_long, aes(x = overall.wr)) +
   geom_histogram(binwidth = 1)  +
   geom_vline(aes(xintercept = mean(overall.wr)),
              color = "red", linewidth = 1)
```



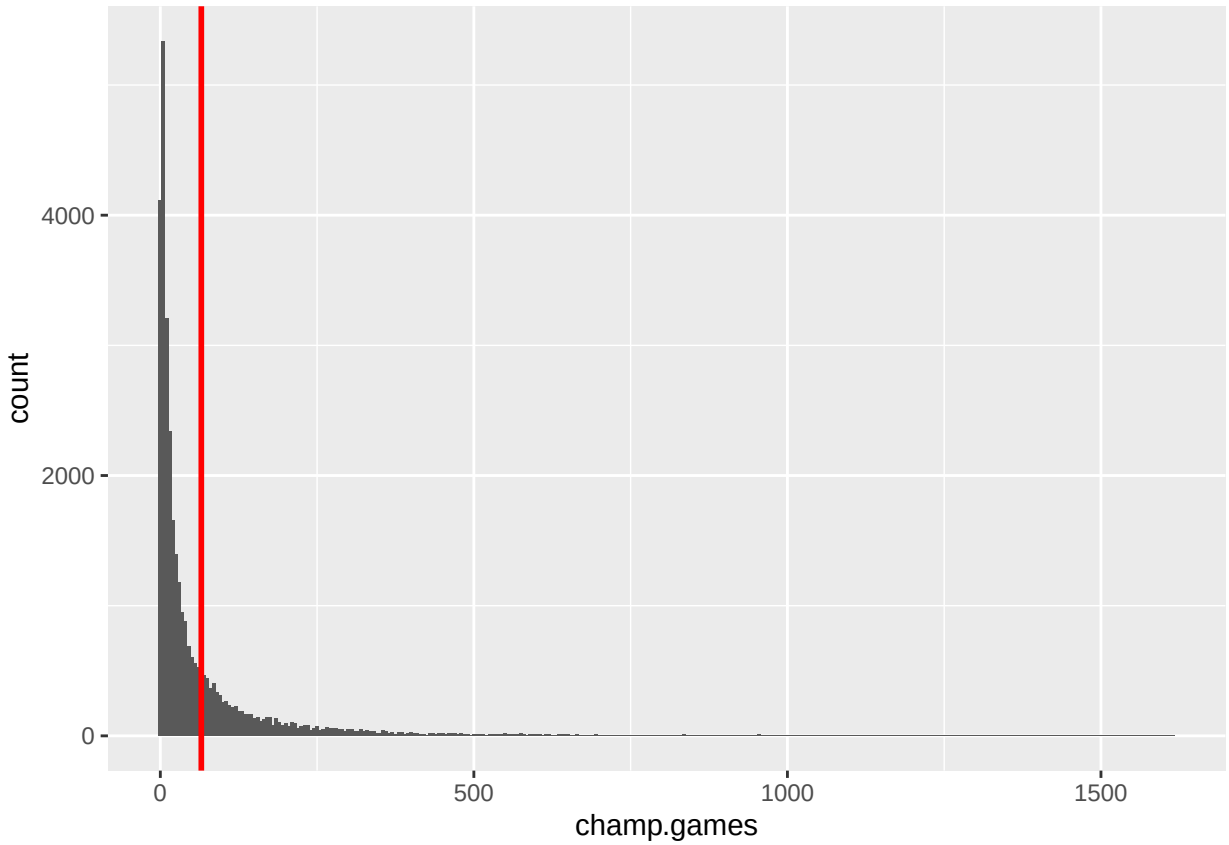

```
ggplot(league_long, aes(x = champ.wr)) +
  geom_histogram(binwidth = 1) +
  geom_vline(aes(xintercept = mean(champ.wr)), color = "red", linewidth = .1)
```



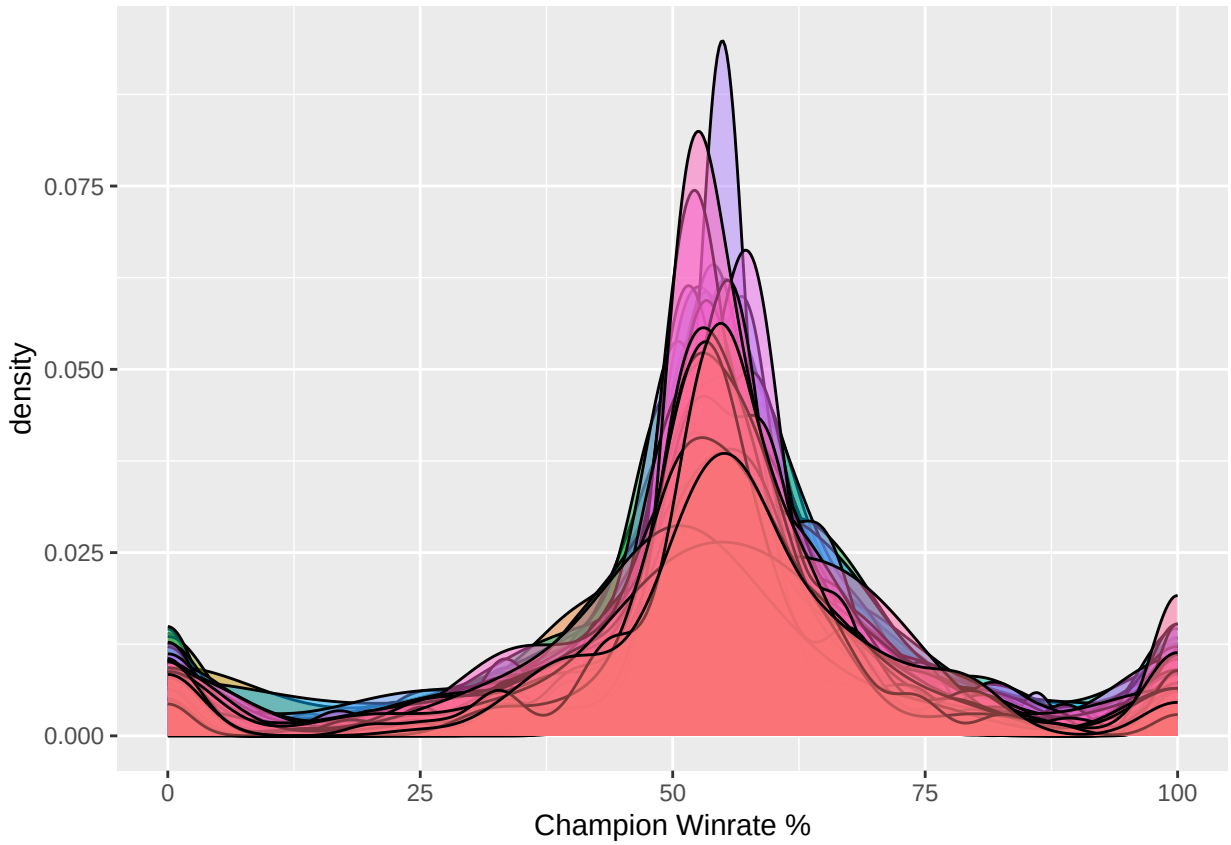

```
ggplot(league_long, aes(x = total.games)) +
   geom_histogram(binwidth = 10)  +
   geom_vline(aes(xintercept = mean(total.games)),
              color = "red", linewidth = 1)
```



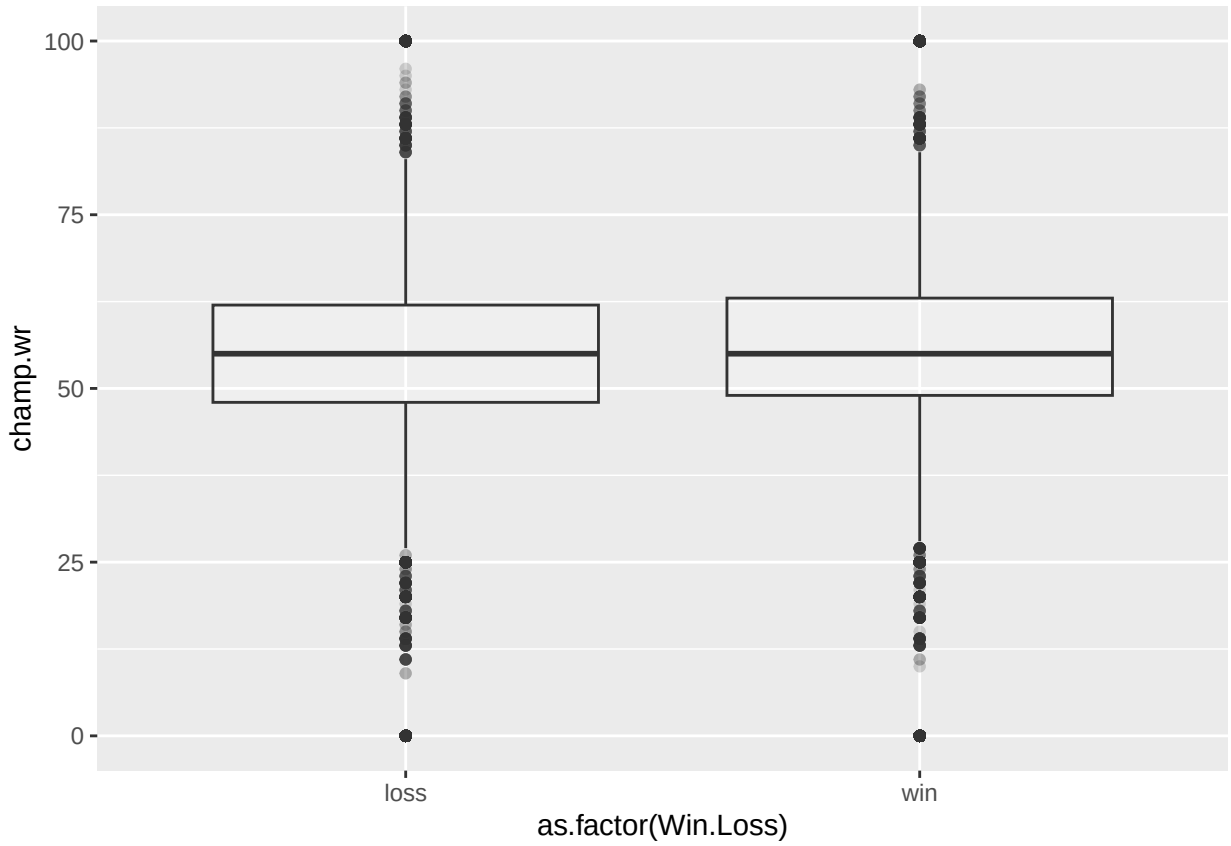

```
ggplot(league_long, aes(x = champ.games)) +
   geom_histogram(binwidth = 5)  +
   geom_vline(aes(xintercept = mean(champ.games)),
              color = "red", linewidth = 1)
```



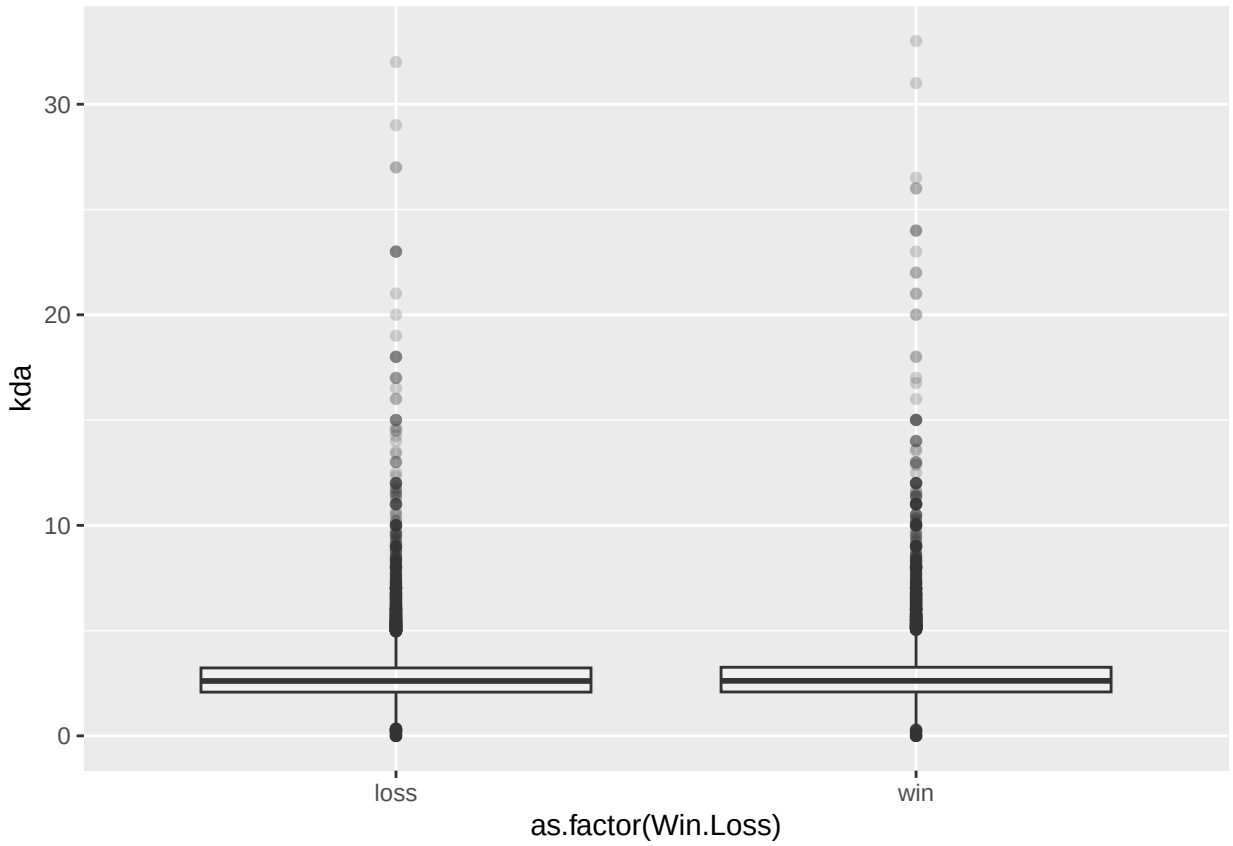

```
#density plot of champion winrates
ggplot(league_long, aes(x = champ.wr)) +
  geom_density(aes(fill = champ.name), alpha = 0.5) +
  xlab("Champion Winrate %")+
  theme(legend.position = "none")
```



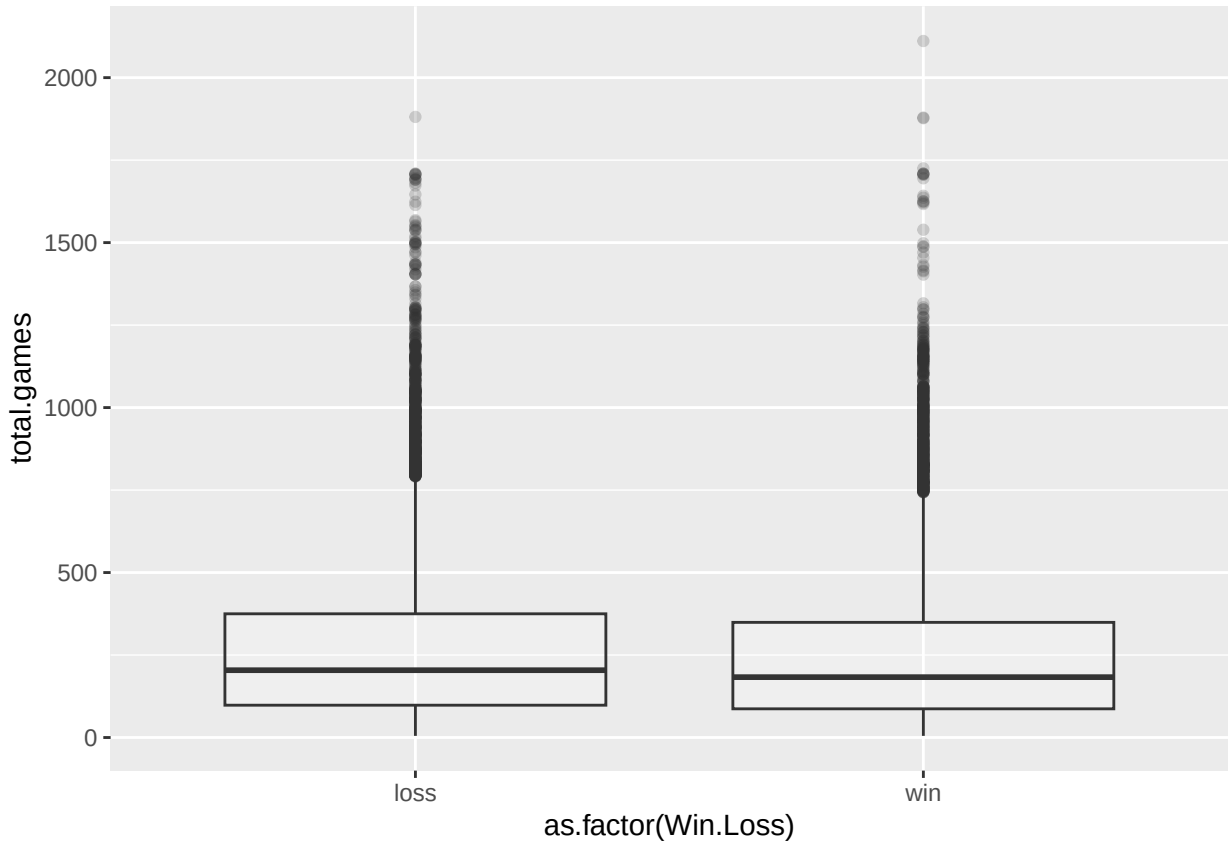

```
#boxplots for each dependent against whether it was a win or loss
ggplot(league_long) +
  geom_boxplot(aes(y = champ.wr, x=as.factor(Win.Loss)), alpha = 0.2)
```



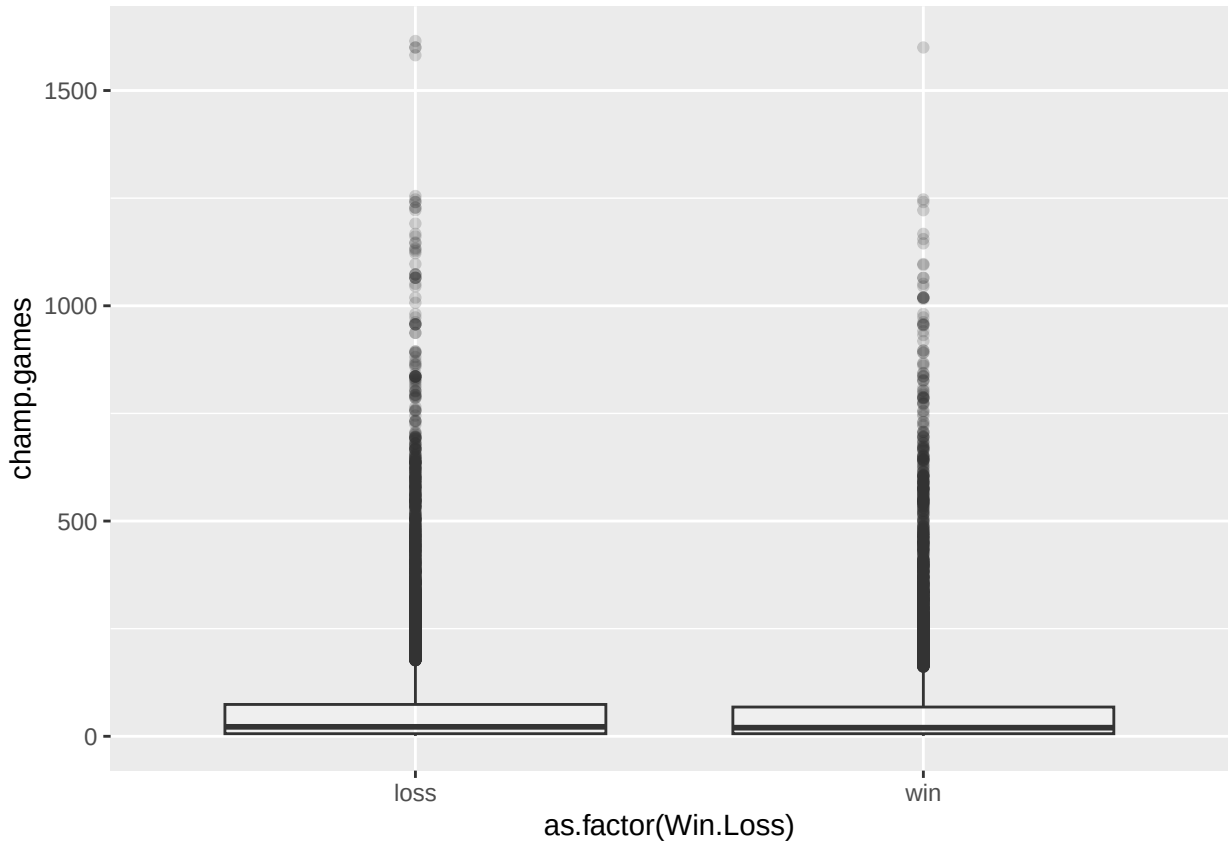

```
ggplot(league_long) +
  geom_boxplot(aes(y = kda, x=as.factor(Win.Loss)), alpha = 0.2)
```



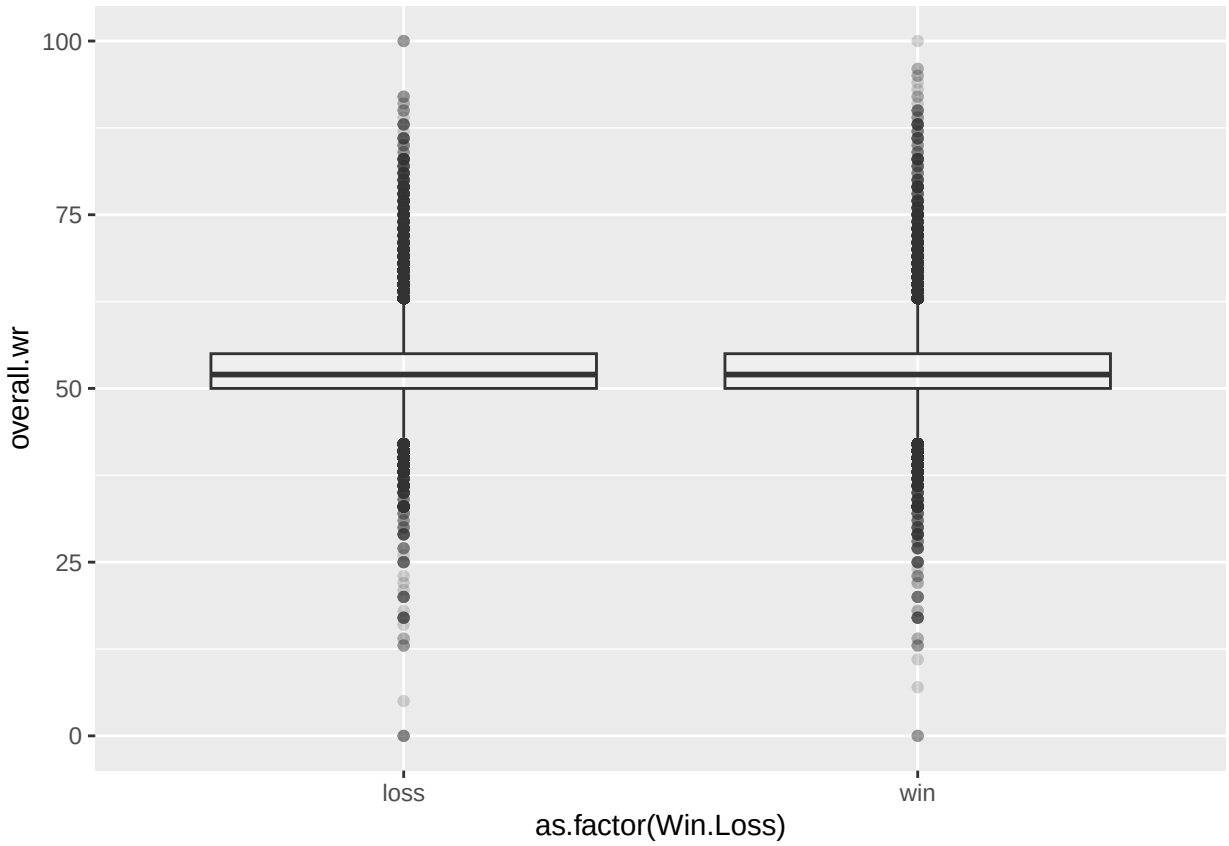

```
ggplot(league_long) +
  geom_boxplot(aes(y = total.games, x=as.factor(Win.Loss)), alpha = 0.2)
```




```
ggplot(league_long) +
  geom_boxplot(aes(y = champ.games, x=as.factor(Win.Loss)), alpha = 0.2)
```




```
ggplot(league_long) +
  geom_boxplot(aes(y = overall.wr, x=as.factor(Win.Loss)), alpha = 0.2)
```



Correlation Scatter

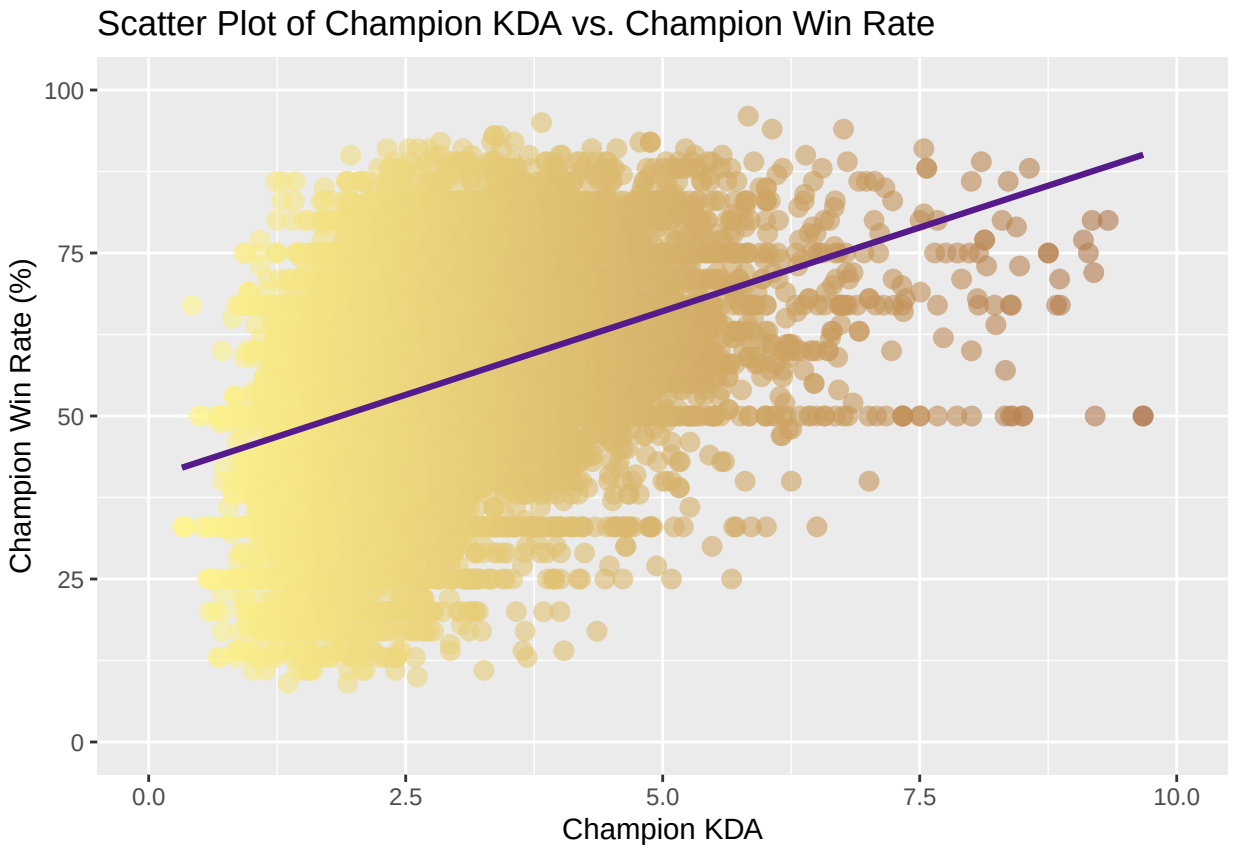
- Creating correlation scatter of KDA and champion winrate
- These variables should be correlated and showed some correlation in corr matrix
- want to see exactly how correlated they are and added a fit line

```
wr_vis_data <- subset(league_long, champ.wr < 100 & champ.wr > 0) #taking out outliers for vis

ggplot(wr_vis_data, aes(x = kda, y = champ.wr)) +
  geom_point(aes(color = kda), size = 3, alpha = 0.6) +  #color is according to kda
  geom_smooth(method = "lm", color = "purple4",se = FALSE, linetype = "solid", linewidth = 1.1) +  #reg
  scale_color_gradient(low = "khaki1", high = "coral4") + #color gradient
  ylim(0, 100) +
  xlim(0, 10) +
  theme(legend.position = "none") +
  labs(title = "Scatter Plot of Champion KDA vs. Champion Win Rate",
       x = "Champion KDA",
       y = "Champion Win Rate (%)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 14 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 14 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



2d Visualization of team statistics

- reading in csv with team statistics that will be used for prediction
- using tsne to see the groupings of the classes
- seeing how effective the features are for predicting

```
library(Rtsne)

team_df <-read.csv("team_average_gold.csv")
head(team_df)
```

```
##   Win.Loss avg_ally_champ_wr avg_ally_champ_games avg_ally_total_games
## 1      win            60.422                 27.8                126.02
## 2     loss            45.400                 38.6                 88.20
## 3     loss            46.800                 63.6                128.00
## 4     loss            29.600                 23.4                213.60
## 5     loss            21.200                  5.2                 74.60
## 6     loss            45.600                 34.6                229.60
##   avg_ally_overall_wr avg_ally_cs avg_ally_kda avg_enemy_champ_wr
## 1              52.812       98.50     3.426720               51.0
## 2              53.400      167.72     2.123740               65.0
## 3              50.400      146.36     1.950620               42.8
## 4              51.000      148.26     1.591762               49.4
## 5              31.200      140.28     1.292240               62.8
## 6              52.600      146.36     1.646260               57.8
##   avg_enemy_champ_games avg_enemy_total_games avg_enemy_overall_wr avg_enemy_cs
## 1                  64.6                 127.2                 44.4       139.98
```

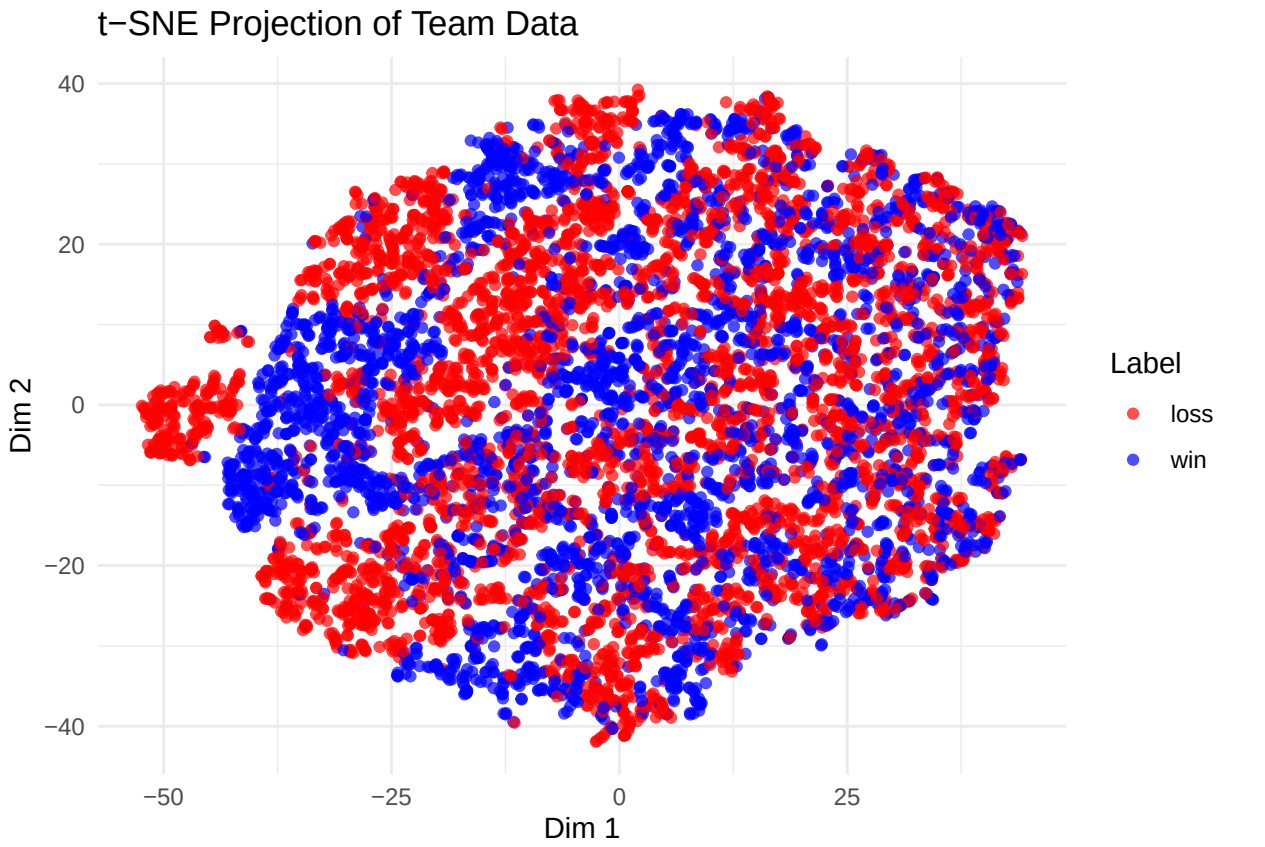
```
## 2                 31.0                113.2                48.8       148.58
## 3                 52.0                175.0                48.4       125.30
## 4                 58.8                144.4                47.8       158.20
## 5                 16.0                 81.6                53.6       136.56
## 6                 25.2                113.8                47.6       139.50
##   avg_enemy_kda
## 1      2.521420
## 2      3.210560
## 3      2.271360
## 4      2.885146
## 5      2.387600
## 6      4.443860
```

```
#double checking theres no duplicates in both dfs
features <- team_df %>% select(-Win.Loss)
features_no_dup <- features[!duplicated(features), ]

#make sure labels are in same order
labels_no_dup <- team_df$Win.Loss[!duplicated(features)]

#use tsne
tsne_result <- Rtsne(as.matrix(features_no_dup), dims = 2, perplexity = 30, verbose = TRUE)
```

```
## Performing PCA
## Read the 7927 x 12 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 1.42 seconds (sparsity = 0.015531)!
## Learning embedding...
## Iteration 50: error is 94.967481 (50 iterations in 0.75 seconds)
## Iteration 100: error is 86.503135 (50 iterations in 0.76 seconds)
## Iteration 150: error is 85.471075 (50 iterations in 0.74 seconds)
## Iteration 200: error is 85.471366 (50 iterations in 0.76 seconds)
## Iteration 250: error is 85.474971 (50 iterations in 0.79 seconds)
## Iteration 300: error is 3.110055 (50 iterations in 0.72 seconds)
## Iteration 350: error is 2.794712 (50 iterations in 0.71 seconds)
## Iteration 400: error is 2.607611 (50 iterations in 0.73 seconds)
## Iteration 450: error is 2.483129 (50 iterations in 0.74 seconds)
## Iteration 500: error is 2.394821 (50 iterations in 0.73 seconds)
## Iteration 550: error is 2.328612 (50 iterations in 0.73 seconds)
## Iteration 600: error is 2.277883 (50 iterations in 0.76 seconds)
## Iteration 650: error is 2.237805 (50 iterations in 0.74 seconds)
## Iteration 700: error is 2.206302 (50 iterations in 0.76 seconds)
## Iteration 750: error is 2.181871 (50 iterations in 0.78 seconds)
## Iteration 800: error is 2.163136 (50 iterations in 0.77 seconds)
## Iteration 850: error is 2.148567 (50 iterations in 0.77 seconds)
## Iteration 900: error is 2.137689 (50 iterations in 0.78 seconds)
## Iteration 950: error is 2.128767 (50 iterations in 0.77 seconds)
## Iteration 1000: error is 2.120735 (50 iterations in 0.76 seconds)
## Fitting performed in 15.08 seconds.
```

```
#put labels on the tsne results
tsne_data <- data.frame(X = tsne_result$Y[,1], Y = tsne_result$Y[,2], Label = labels_no_dup)
```

```r
#plot
ggplot(tsne_data, aes(x = X, y = Y, color = Label)) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  labs(title = "t-SNE Projection of Team Data", x = "Dim 1", y = "Dim 2") +
  scale_color_manual(values = c("win" = "blue", "loss" = "red"))
```



Conclusion of eda

- From this eda we can see variables such as champ wr, total games, champ games, and kda have more correlation with win or loss than the other vars
- From the visualizations in different dimensions, we can see the classes are seperable
  - Even though there are lots of seperable chunks there are still lots of overlapping samples
  - To best deal with the patterns in the data a good predictor to use will be random forest since it is decision tree based and can find complex non linear relationships. It will be able to map the more complex relationship between features and the target variable