R Notebook

```
#setwd("~/R projects/League")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(skimr)
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip

## -- Attaching packages --------------------------------------- tidymodels 0.1.3 --

## v broom        0.7.9      v rsample      0.1.0
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.3
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.8
## v recipes      0.1.16

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```
data <- read_csv("league.csv")
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 129852 Columns: 119

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (15): gameid, datacompleteness, url, league, split, side, position, pl...
## dbl  (100): year, playoffs, game, patch, playerid, gamelength, result, kills...
## lgl    (3): dragons (type unknown), turretplates, opp_turretplates
## dttm   (1): date

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(data)
```

```
## # A tibble: 6 x 119
##   gameid  datacompleteness url   league  year split playoffs date
##   <chr>   <chr>            <chr> <chr>  <dbl> <chr>    <dbl> <dttm>
## 1 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## 2 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## 3 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## 4 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## 5 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## 6 ESPORT~ complete         http~ KeSPA   2021 <NA>         0 2021-01-02 07:40:39
## # ... with 111 more variables: game <dbl>, patch <dbl>, playerid <dbl>,
## #   side <chr>, position <chr>, player <chr>, team <chr>, champion <chr>,
## #   ban1 <chr>, ban2 <chr>, ban3 <chr>, ban4 <chr>, ban5 <chr>,
## #   gamelength <dbl>, result <dbl>, kills <dbl>, deaths <dbl>, assists <dbl>,
## #   teamkills <dbl>, teamdeaths <dbl>, doublekills <dbl>, triplekills <dbl>,
## #   quadrakills <dbl>, pentakills <dbl>, firstblood <dbl>,
## #   firstbloodkill <dbl>, firstbloodassist <dbl>, firstbloodvictim <dbl>, ...
```

I am not interested in the data of every region. I only want the data from NA, EU, KR and CH

```
table(data$league)
```

```
##
##     AOL     BIG      BL      BM   CBLOL CBLOL.A      CU      DL     EBL     EGL
##     444    1776    1116    1056    2640    2652    2448    1092    1692    1104
##      EM     GLL     GSG      HC      HM   KeSPA     LCK  LCK CL     LCL     LCO
##    3168    2184     636    2112    2148      36    5724    1440    1776    2196
##     LCS   LCS.A     LDL     LEC     LFL     LHE     LJL     LLA     LMF     LPL
##    4140    3996   16332    2964    2604    3456    1860    2160    3840    8724
##   LPLOL LVP DDH     MSI    NERD     NLC   OTBLX     PCS     PGN     PRM     RCL
##    1668    2580     972     504    2724    3732    3192    2220    2580    4212
##      SL     TCL     TRA     UGP    UKLC      UL     UPL     VCS     WCS
##    2724    2688    2448    1584    1812    2028    2352    2112     204
```

```
leagues <- c("LPL", "LCS", "LCK", "LEC")
data <- data %>%
  filter(league %in% leagues)
```

```
table(data$datacompleteness)
```

```
##
## complete  partial
##    16968     4584
```

```
names(data)
```

```
##   [1] "gameid"                 "datacompleteness"
##   [3] "url"                    "league"
##   [5] "year"                   "split"
##   [7] "playoffs"               "date"
##   [9] "game"                   "patch"
##  [11] "playerid"               "side"
##  [13] "position"               "player"
##  [15] "team"                   "champion"
##  [17] "ban1"                   "ban2"
##  [19] "ban3"                   "ban4"
##  [21] "ban5"                   "gamelength"
##  [23] "result"                 "kills"
##  [25] "deaths"                 "assists"
##  [27] "teamkills"              "teamdeaths"
##  [29] "doublekills"            "triplekills"
##  [31] "quadrakills"            "pentakills"
##  [33] "firstblood"             "firstbloodkill"
##  [35] "firstbloodassist"       "firstbloodvictim"
##  [37] "team kpm"               "ckpm"
##  [39] "firstdragon"            "dragons"
##  [41] "opp_dragons"            "elementaldrakes"
##  [43] "opp_elementaldrakes"    "infernals"
##  [45] "mountains"              "clouds"
##  [47] "oceans"                 "dragons (type unknown)"
##  [49] "elders"                 "opp_elders"
```

```
##  [51] "firstherald"              "heralds"
##  [53] "opp_heralds"              "firstbaron"
##  [55] "barons"                   "opp_barons"
##  [57] "firsttower"               "towers"
##  [59] "opp_towers"               "firstmidtower"
##  [61] "firsttothreetowers"       "turretplates"
##  [63] "opp_turretplates"         "inhibitors"
##  [65] "opp_inhibitors"           "damagetochampions"
##  [67] "dpm"                      "damageshare"
##  [69] "damagetakenperminute"     "damagemitigatedperminute"
##  [71] "wardsplaced"              "wpm"
##  [73] "wardskilled"              "wcpm"
##  [75] "controlwardsbought"       "visionscore"
##  [77] "vspm"                     "totalgold"
##  [79] "earnedgold"               "earned gpm"
##  [81] "earnedgoldshare"          "goldspent"
##  [83] "gspd"                     "total cs"
##  [85] "minionkills"              "monsterkills"
##  [87] "monsterkillsownjungle"    "monsterkillsenemyjungle"
##  [89] "cspm"                     "goldat10"
##  [91] "xpat10"                   "csat10"
##  [93] "opp_goldat10"             "opp_xpat10"
##  [95] "opp_csat10"               "golddiffat10"
##  [97] "xpdiffat10"               "csdiffat10"
##  [99] "killsat10"                "assistsat10"
## [101] "deathsat10"               "opp_killsat10"
## [103] "opp_assistsat10"          "opp_deathsat10"
## [105] "goldat15"                 "xpat15"
## [107] "csat15"                   "opp_goldat15"
## [109] "opp_xpat15"               "opp_csat15"
## [111] "golddiffat15"             "xpdiffat15"
## [113] "csdiffat15"               "killsat15"
## [115] "assistsat15"              "deathsat15"
## [117] "opp_killsat15"            "opp_assistsat15"
## [119] "opp_deathsat15"
```

Now I am going to split the data into 2 datasets. One with player data and one with team data.

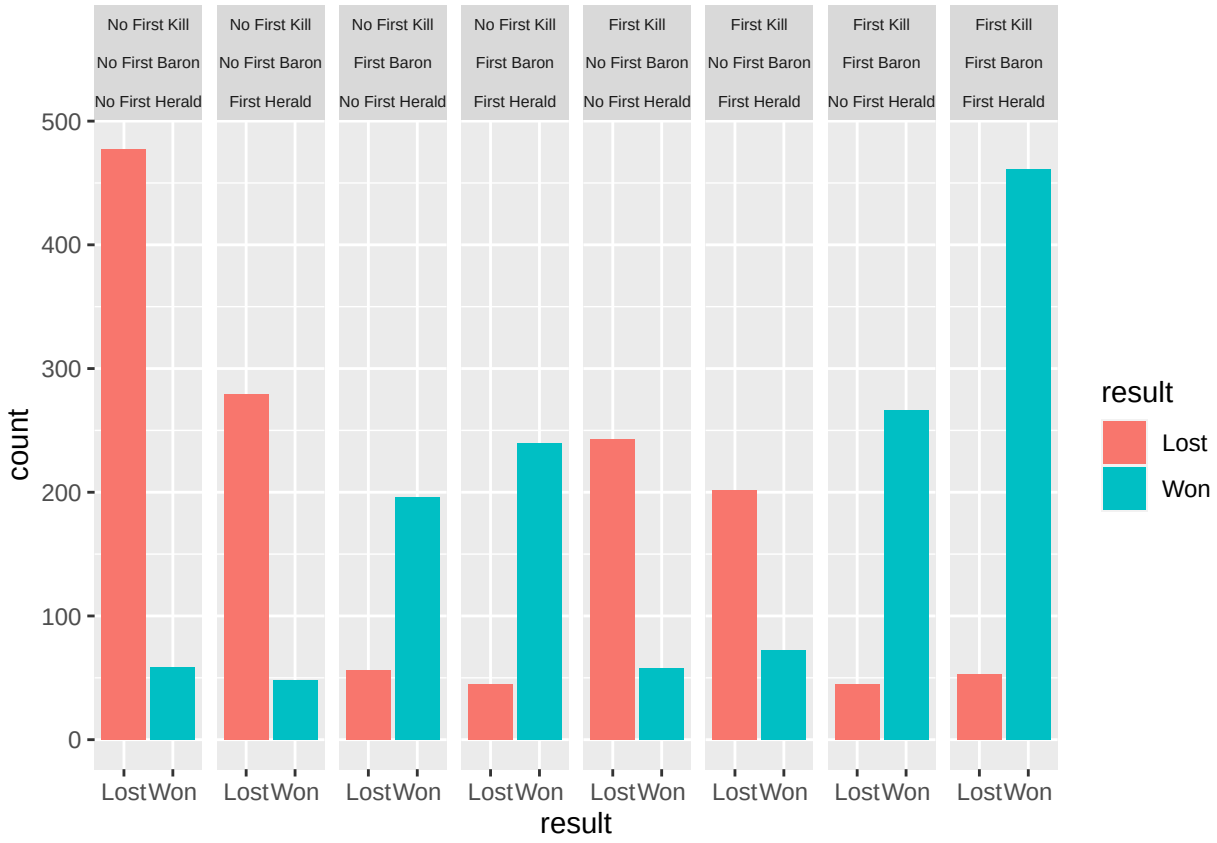
```
team_data <- data %>%
  filter(position == "team")

player_data <- data %>%
  filter(position != "team")
```

Lets fix some of the columns in team_data

```
team_data$result <- factor(team_data$result, labels = c("Lost", "Won"))
team_data$firstbaron <- factor(team_data$firstbaron, labels = c( "No First Baron", "First Baron"))
team_data$firstblood <- factor(team_data$firstblood, labels = c("No First Kill", "First Kill"))
team_data$firstdragon <- factor(team_data$firstdragon, labels = c("No First Dragon", "First Dragon"))
team_data$firstherald <- factor(team_data$firstherald, labels = c("No First Herald", "First Herald"))
team_data$firstmidtower <- factor(team_data$firstmidtower, labels = c( "No First Midtower", "First Midt
```

Now lets see how getting this objective affect the chances of winning the game.

```
ggplot(data = team_data %>% filter(!is.na(firstblood), !is.na(firstbaron), !is.na(firstherald))
       , aes(x = result, fill = result)) +
  geom_bar() +
  facet_grid(~ firstblood + firstbaron + firstherald, scale = "free") +
  theme(strip.text.x = element_text(size = 6.375))
```



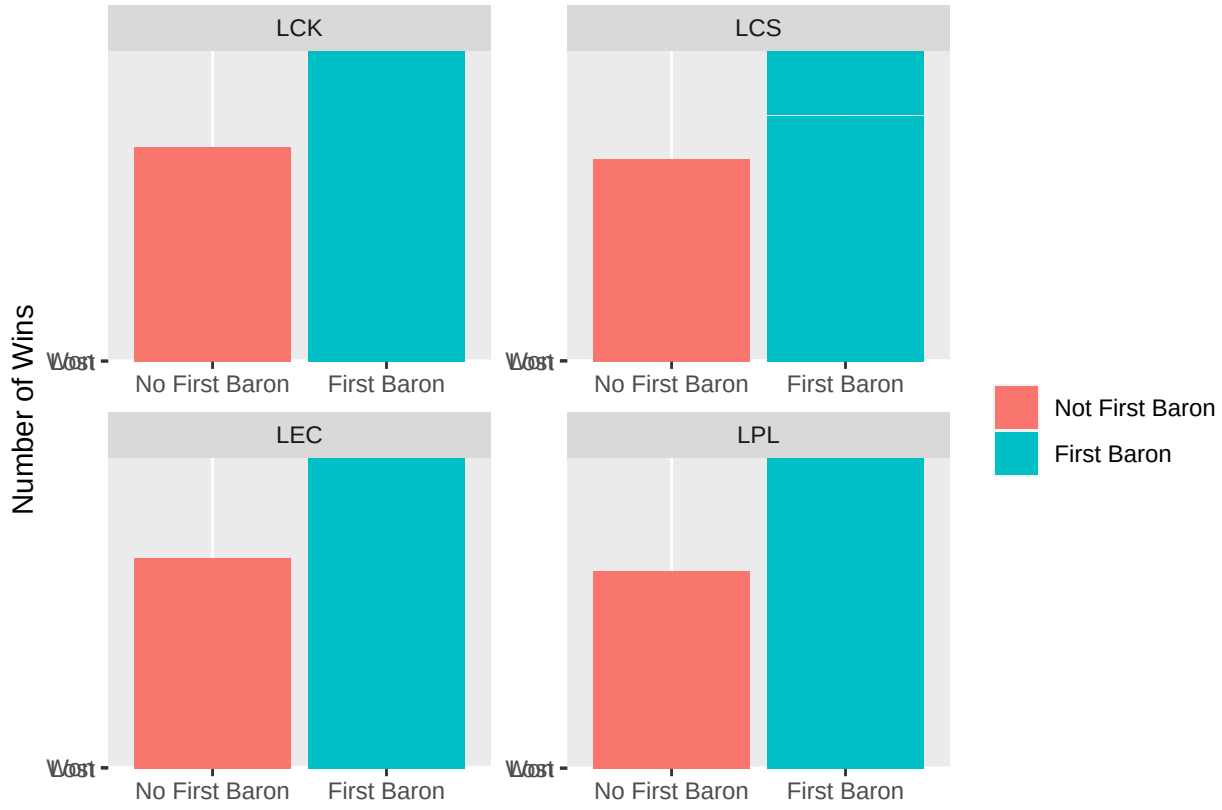
Definitely seems like these factors decide the outcome of the game. The first baron seem to be the most important facctor of them. It also seems like they are quite correlated since the number of games where one team has all them is quite a lot more than the number of game where they are split between teams.

Lets see if the getting firsttblood has a different effect on winning the game depending on the region and also get an idea on how it affects the outcome of a game by itself.

```
ggplot(data = team_data %>% filter(!is.na(firstblood)),
       aes(y = result,
           x = factor(firstblood),
           fill = factor(firstblood))) +
  geom_bar(stat = "identity") +
  scale_x_discrete(labels = c("No Firstblood", "Firstblood")) +
  scale_fill_discrete("",label = c("No Firstblood", "Firstblood")) +
  xlab("") +
  facet_wrap(~league, scales = "free") +
  ylab("Number of wins")
```

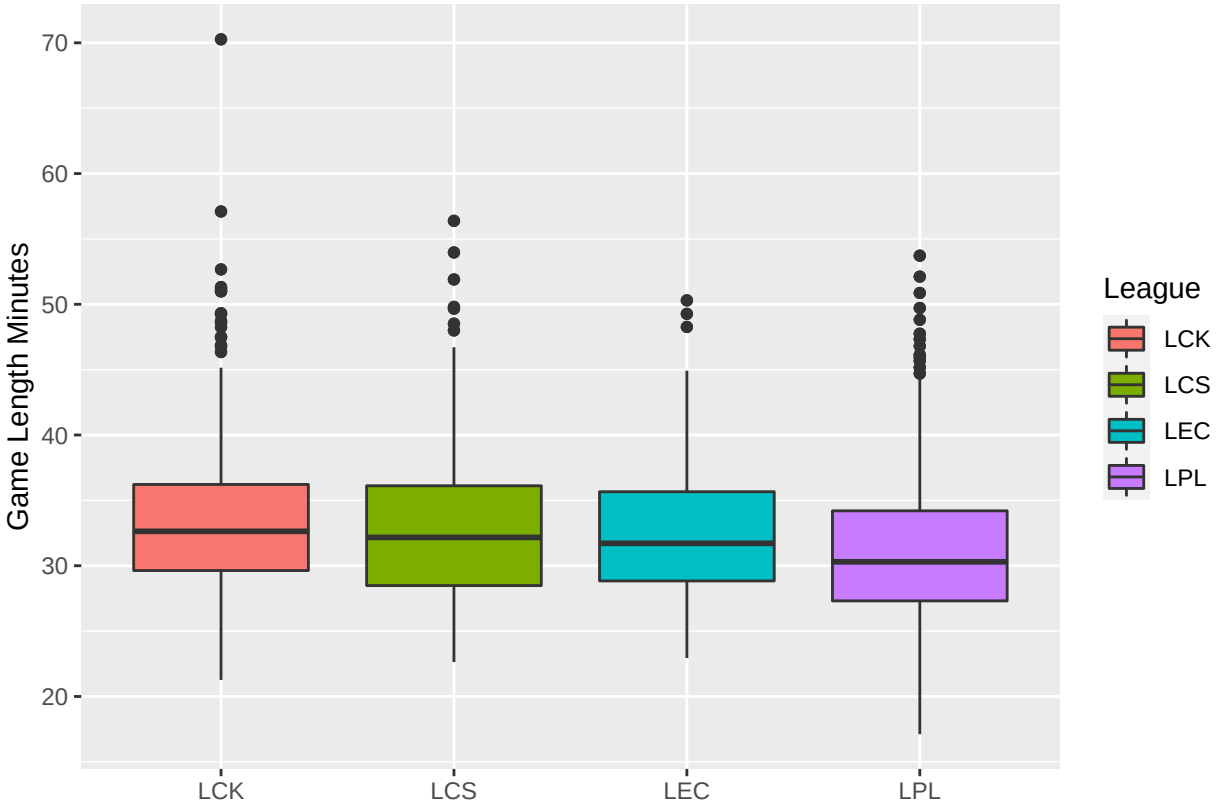
Lets check the first baron's affect on the chances of winning by region.

```
ggplot(data = team_data %>% filter(!is.na(firstbaron)),
       aes(x = factor(firstbaron),
           y = result,
           fill = factor(firstbaron))) +
  geom_bar(stat = "identity") +
  ylab("Number of Wins") +
  xlab("") +
  scale_fill_discrete("", label = c("Not First Baron", "First Baron")) +
  facet_wrap(~league, scale = "free")
```



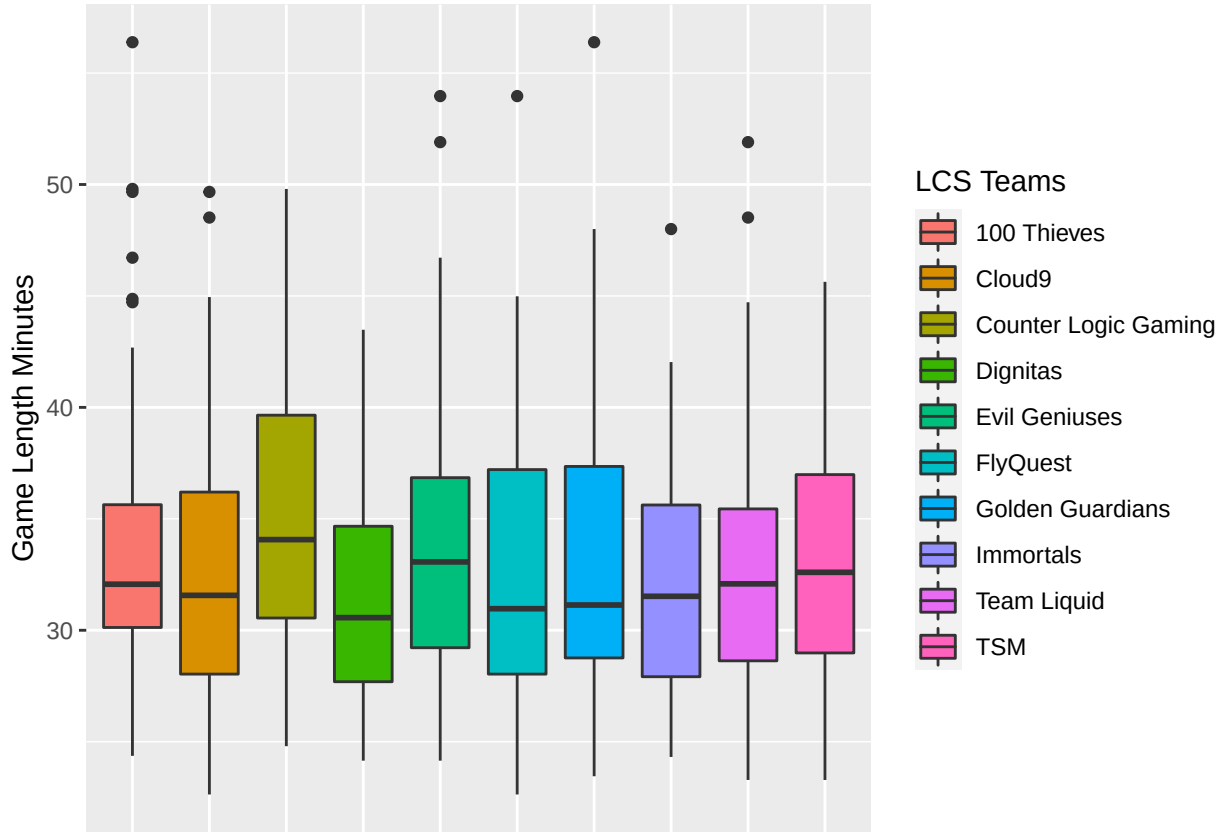
Here, I want to look at if there is a difference in game length for the different regions.

```
ggplot(data = team_data, aes(x = factor(league), y = gamelength/60, fill = factor(league))) +
  geom_boxplot() +
  ylab("Game Length Minutes") +
  xlab("") +
  scale_fill_discrete("League")
```



Lets look at the difference in game length between the teams in the LCS.

```
ggplot(data = team_data %>% filter(league == "LCS"), aes(x = factor(team), y = gamelength/60, fill = fa
  geom_boxplot() +
  ylab("Game Length Minutes") +
  xlab("") +
  scale_fill_discrete("LCS Teams") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```



Now when I have looked at some of the variables in the data set and I want to try to predict the outcomes of the games given the data.

First, I split the data into a training and testing set so that I can test the performance of the model

```
split <- initial_split(team_data, prop = 0.8, strata = result)
train <- training(split)
test <- testing(split)
```

First I will create a recipe about how to deal with the data and which variables I want to use. The only preprocessing required is removing rows that contains missing values which is fine since I have so many samples.

```
team_rec <- recipe(result ~ firstbaron + firstblood + firstherald + firstmidtower + firstdragon,
                   data = team_data) %>%
  step_naomit(all_predictors()) %>%
  step_naomit(all_outcomes(), skip = TRUE)
```

For creating a model I am looking for an interpretable model so that I can understand the data better. Therefore, I am going to use a logistic regression model.

I am not going to try to fix the parameters of the model.

```
log_model <- logistic_reg()
```

```
team_wf <- workflow() %>%
  add_model(log_model) %>%
  add_recipe(team_rec)
```

Since we are removing rows that contain NA we need to have a preproccesed test set to compare our results with

```
log_rec_prep <- prep(team_rec)
test1 <- bake(log_rec_prep, new_data = test)
```

```
log_fit <- fit(team_wf, data = train)
log_pred <- predict(log_fit, new_data = test)
confusionMatrix(data = log_pred$.pred_class,
                reference = test1$result)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Lost Won
##       Lost  234  64
##       Won    37 230
##
##                Accuracy : 0.8212
##                  95% CI : (0.7871, 0.852)
##     No Information Rate : 0.5204
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6433
##
##  Mcnemar's Test P-Value : 0.009679
##
##             Sensitivity : 0.8635
##             Specificity : 0.7823
##          Pos Pred Value : 0.7852
##          Neg Pred Value : 0.8614
##              Prevalence : 0.4796
##          Detection Rate : 0.4142
##    Detection Prevalence : 0.5274
##       Balanced Accuracy : 0.8229
##
##        'Positive' Class : Lost
##
```

Here we see that with this simple model with very few variables we can get good results. Lets see how our model used the variables to predict.

```
log_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
```

```
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_naomit()
## * step_naomit()
##
## -- Model ----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##                  (Intercept)        firstbaronFirst Baron
##                      -2.6665                      3.2170
##        firstbloodFirst Kill      firstheraldFirst Herald
##                       0.1943                     -0.0566
## firstmidtowerFirst Midtower      firstdragonFirst Dragon
##                       1.6132                      0.3998
##
## Degrees of Freedom: 2234 Total (i.e. Null);  2229 Residual
## Null Deviance:       3098
## Residual Deviance: 1693  AIC: 1705
```

Here we see that the model says that first baron was the most valued predictor for the result of the game and after that first mid tower was the most valued predictor. We can also see that first herald is not a valued predictor