



eSports Analytics: League of Legends Optimization



Authors:

Ning Shi

[nshi02@syr.edu](mailto:nshi02@syr.edu)

Tanbir Biryajh

[tsbiryaj@syr.edu](mailto:tsbiryaj@syr.edu)

Rashad Davis

[rkdavis@syr.edu](mailto:rkdavis@syr.edu)

Graham Latsa

[gmlatsa@syr.edu](mailto:gmlatsa@syr.edu)

Table of Contents

[Table of Contents 1](#_Toc532913352)

[Table of Figures 3](#_Toc532913353)

[Introduction 4](#_Toc532913354)

[Opportunity 4](#_Toc532913355)

[Obtaining the Dataset 5](#_Toc532913356)

[Data Overview 6](#_Toc532913357)

[Description of Dataset 6](#_Toc532913358)

[Data Acquisition, Load, & Transformation 6](#_Toc532913359)

[Data Architecture 7](#_Toc532913360)

[Data Cleansing 7](#_Toc532913361)

[Data Quality 7](#_Toc532913362)

[Business Perspective 8](#_Toc532913363)

[Teams Role 8](#_Toc532913364)

[Business Questions 8](#_Toc532913365)

[Data Analysis with Visualization & Interpretation 9](#_Toc532913366)

[Answering Question #1 9](#_Toc532913367)

[Answering Question #2 10](#_Toc532913368)

[Answering Question #3 13](#_Toc532913369)

[Answering Question #4 14](#_Toc532913370)

[Answering Question #5 15](#_Toc532913371)

[Answering Question #6 16](#_Toc532913372)

[Answering Question #7 17](#_Toc532913373)

[Answering Question #8 19](#_Toc532913374)

[Answering Question #9 20](#_Toc532913375)

[Answering Question #10 21](#_Toc532913376)

[Answering Question #11 22](#_Toc532913377)

[Answering Question #12 24](#_Toc532913378)

[Answering Question $13 27](#_Toc532913379)

[Answering Question #14 28](#_Toc532913380)

[Answering Question #15 29](#_Toc532913381)

[Answering Question #16 30](#_Toc532913382)

[Conclusions - Actions Insights 31](#_Toc532913383)

[Review 31](#_Toc532913384)

[Strategy 31](#_Toc532913385)

[Champions 31](#_Toc532913386)

[Teams 31](#_Toc532913387)

[Appendix - R code 32](#_Toc532913388)

[Environment configuration 32](#_Toc532913389)

[Business Question #2 – R-code 34](#_Toc532913390)

[Business Question #3 – R-code 36](#_Toc532913391)

[Business Question #4 – R-code 39](#_Toc532913392)

[Business Question #5 – R-code 41](#_Toc532913393)

[Business Question #6 – R-code 44](#_Toc532913394)

[Business Question #7 – R-code 46](#_Toc532913395)

[Business Question # 8 – R-code 48](#_Toc532913396)

[Business Question #9 – R-code 49](#_Toc532913397)

[Business Question #10 – R-code 50](#_Toc532913398)

[Business Question #11– R-code 53](#_Toc532913399)

[Business Question #12 – R-code 58](#_Toc532913400)

[Business Question #13 – R-code 61](#_Toc532913401)

[Business Question #14 – R-code 62](#_Toc532913402)

[Business Question #15 – R-code 63](#_Toc532913403)

[Business Question #16 – R-code 64](#_Toc532913404)

Table of Figures

[Figure 1 Goldman Sachs Ranking Top eSports Platform & Goldman Sachs Rank eSports vs Sports ↑ 4](#_Toc532913312)

[Figure 2 League of Legends Popularity vs. Sports ↑ 5](#_Toc532913313)

[Figure 3 League of Legends Tournaments and Prize Money ↑ 5](#_Toc532913314)

[Figure 4 Top Five Champions ↑ 10](#_Toc532913315)

[Figure 5 Bottom Five Champions ↑ 10](#_Toc532913316)

[Figure 6 Win Rate per Champion ↑ 11](#_Toc532913317)

[Figure 7 Win Rate per Above Average Champion ↑ 11](#_Toc532913318)

[Figure 8 Win Rate per Below Average Champion ↑ 11](#_Toc532913319)

[Figure 9 Win Rate for the Top 25% Champions ↑ 12](#_Toc532913320)

[Figure 10 Win Rate for the Bottom 25% Champions ↑ 12](#_Toc532913321)

[Figure 11 Top Champions that beat Janna ↑ 13](#_Toc532913322)

[Figure 12 Summoner Spell Preference for Janna ↑ 14](#_Toc532913323)

[Figure 13 Bar Chart for Top 10 Most Popular Champions ↑ 15](#_Toc532913324)

[Figure 14 Bar Chart: Ban Rate for Champions ↑ 16](#_Toc532913325)

[Figure 15 Bar Chart: Top 10 Banned Chaps with Win Rates ↑ 16](#_Toc532913326)

[Figure 16 Game Winner by Team 1 Tower Kills ↑ 17](#_Toc532913327)

[Figure 17 Generalized Linear Model Summary ↑ 17](#_Toc532913328)

[Figure 18 VIF Test to test for collinearity ↑ 17](#_Toc532913329)

[Figure 19 Logit Analysis for Prediction ↑ 18](#_Toc532913330)

[Figure 20 Logit Model Confusion Matrix to Calculate Accuracy ↑ 18](#_Toc532913331)

[Figure 21 Probability of Winning Team as Demonstrated by t1\_towerkills ↑ 18](#_Toc532913332)

[Figure 22 Pie Chart of both Team Win Rate ↑ 19](#_Toc532913333)

[Figure 23 Linear Model Summary of First Kill Objectives ↑ 20](#_Toc532913334)

[Figure 24 Prediction using Linear Model and First Kill Objectives ↑ 20](#_Toc532913335)

[Figure 25 Top 10 Team 1 Compositions with the Most Objective Kills ↑ 21](#_Toc532913336)

[Figure 26 Top 10 Team 2 Compositions with the Most Objective Kills ↑ 21](#_Toc532913337)

[Figure 27 Total Summoner Spell Frequency Choice of All Players ↑ 22](#_Toc532913338)

[Figure 28 Yasuo Win Rate Based on Summoner Spell ↑ 22](#_Toc532913339)

[Figure 29 Janna Win Rate Based on Summoner Spell ↑ 23](#_Toc532913340)

[Figure 30 Team 1 Compiled Objectives vs Average Team 2 Compiled Objectives ↑ 24](#_Toc532913341)

[Figure 31 ksvm Plot – Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑ 25](#_Toc532913342)

[Figure 32 ksvm Confusion Matrix and Accuracy ↑ 25](https://sumailsyr-my.sharepoint.com/personal/tsbiryaj_syr_edu/Documents/IST687%20Project-Team%20A2-Final%20Paper-DRAFT-17Dec2018.docx#_Toc532913343)

[Figure 33 Naïve Bayes Plot– Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑ 26](#_Toc532913344)

[Figure 34 Naïve Bayes Confusion Matrix and Accuracy ↑ 26](https://sumailsyr-my.sharepoint.com/personal/tsbiryaj_syr_edu/Documents/IST687%20Project-Team%20A2-Final%20Paper-DRAFT-17Dec2018.docx#_Toc532913345)

[Figure 35 Model Comparison - Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑ 26](#_Toc532913346)

[Figure 36 Team 1 and Team 2 Win Percentages given Rift Herald Kill ↑ 27](#_Toc532913347)

[Figure 37 Win Percentages given Rift Herald Kill ↑ 27](#_Toc532913348)

[Figure 38 Testing the nemesis function ↑ 28](#_Toc532913349)

[Figure 39 Testing the champspellcombo function ↑ 29](#_Toc532913350)

[Figure 40 Testing the champpartner function ↑ 30](#_Toc532913351)

Introduction

Opportunity

Spacewar!, a video game developed in the early 60s, “involves two spaceships, each controlled by a separate player, attempting to shoot one another while maneuvering on a two-dimensional plane in the gravity well”. This concept seems so simple now, but how this game influenced game developers established this game as one of the “top ten most important video games of all time” (eventually being archived at the library of congress). Despite its accolades for video game development, it additionally is a cultural pillar for the video game industry. This is because the first official eSports tournament (1972) was a Spacewar! competition. It was called the “Intergalactic spacewar olympics" with a winning prize of a 1-year subscription for Rolling Stone.

All light-hearted matters aside, the eSport industry is currently exploding. Read a few of the statements and charts below by leading researchers/investors:

Approximately 300 million people worldwide tune in to eSports today, and that number is growing rapidly. By 2020, that number will be closer to 500 million.

The market is presently undervalued and has significant room to grow.

The audience is high-value and global, and its numbers are rising.

- Business Insider

In 2018, we estimate the global monthly audience for eSports will reach 167mn people, larger than that of Major League Baseball and the National Hockey League.

There are over 2.2bn active gamers globally. Today, the eSports audience represents just 5% of the total online gaming population, which suggests that there should be plenty more runway for eSports audience growth.

We are seeing increased venture investment in the space. Skillz is a platform for mobile eSports that raised $25mn of venture funding in December 2017.

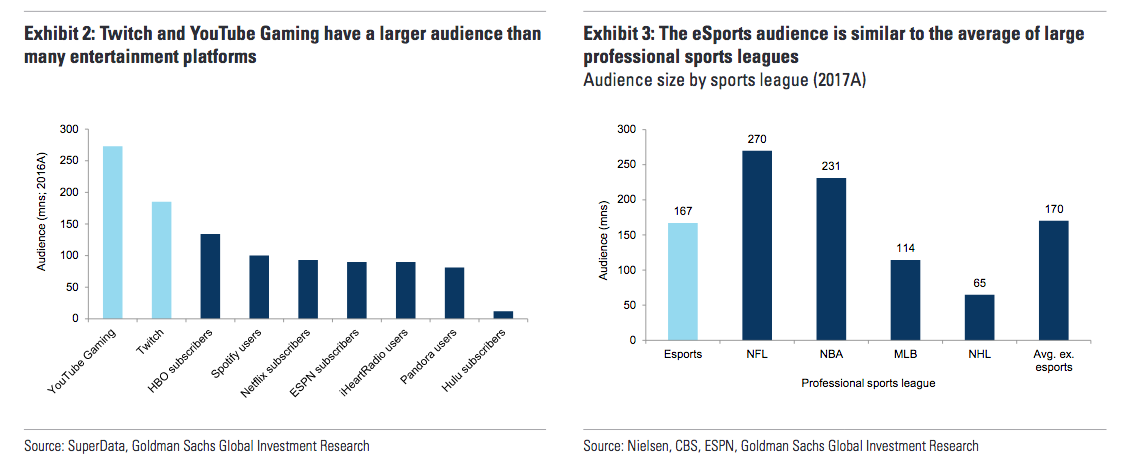


Figure 1 Goldman Sachs Ranking Top eSports Platform & Goldman Sachs Rank eSports vs Sports ↑

- Goldman Sachs

Those were just a few of the statements those power house organizations communicated about the opportunity the eSport Market holds. The team will no longer discuss this information, however if you do want to read more information about the opportunity the eSport industry, you can go to these links, [https://www.goldmansachs.com/insights/pages/infographics/e-sports/report.pdf](https://www.goldmansachs.com/insights/pages/infographics/e-sports/report.pdf%20) & <https://www.businessinsider.com/esports-market-growth-ready-for-mainstream-2017-3> & [https://www.statista.com/topics/3121/esports-market/](https://www.statista.com/topics/3121/esports-market/%20%20)

Obtaining the Dataset

Researching data sets, the team stumbled upon a League of Legends data set on Kaggle, <https://www.kaggle.com/datasnaek/league-of-legends/home.> The data contained meet all our variable and observation requirements. The team then decided it was popular enough of a game, see charts below. Therefore, the team decided to use this data as the object our analytics to determine how we can analyze this data to help an organization in this industry.

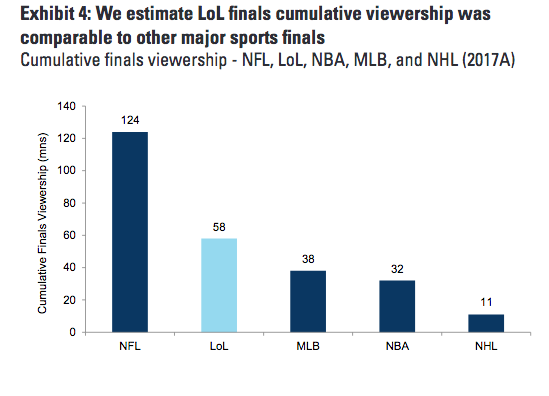


Figure 2 League of Legends Popularity vs. Sports ↑



Figure 3 League of Legends Tournaments and Prize Money ↑

Data Overview

Description of Dataset

The dataset contains data from over 50,000 ranked games from the video game League of Legends. As per publisher Mitchel J, the data was obtained from the Riot Games API. The following variables were captured in the games.csv file:

Game ID

Creation Time (in Epoch format)

Game Duration (in seconds)

Season ID

Winner (1 = Team 1 & 2 = Team 2)

First Baron, dragon, tower, blood, inhibitor and Rift Herald (1 = Team 1 & 2 = Team 2 & 0 = Neither)

Champions and summoner spells for each team (champion and summoner spell IDs are used as keys)

Two additional JSON files were provided to link the champion and summoner spell IDs to their names

Number of tower, inhibitor, Baron, dragon and Rift Herald kills each team had in a game

5 bans of each team (Champion IDs are used as q key)

The Champion JSON file contains the following variables:

Champion Tags

Champion Title

Champion ID

Champion Key

Champion Name

The Summoner Spell JSON file contains the following variables:

Summoner ID

Summoner Level

Summoner Key

Summoner Name

Summoner Description

[https://www.kaggle.com/datasnaek/league-of-legends/home](https://www.kaggle.com/datasnaek/league-of-legends/home%20)

Data Acquisition, Load, & Transformation

League of Legends dataset was accessed through: <https://www.kaggle.com/datasnaek/league-of-legends>. The games dataset was a csv file which was read into R using the read.csv function. The champion and summoner spell files were in the JSON format. Using the jsonlite package, the files were converted from JSON to an R list to an R data frame.

Data Architecture

Below is how our data is composed

* ~ 3MB of Data
* 51,490 Ranked Games ( Observations)
* CSV & JSON Files
* 61 Variables
* Type of data, contains Binary, Factor, Numerical, Char

Data Cleansing

The only data cleansing needed was the team winning column, this is because data was inputted as 1 & 2, and since the data was still binary we adjusted and made it 0 & 1.

Data Quality

The is.na function was used to check our dataset for NAs. The result was there are no NAs in the original dataset. For business question 9, NAs were introduced when creating the T1CompiledObjectives and T2CompiledObjectives. The variables were only created for cases where Team 1 or Team 2 were the winners respectively. Alternative approach for this question would be to use indexing to sort the compiled objectives variables based on if Team 1 was the winner and if Team 2 was the winner. When creating the models for business question 12, the NAs were removed from T1CompiledObjectives and T2CompiledObjectives variables. Values were needed for both variables in order to predict the winner.

Business Perspective

Teams Role

When the team obtained the data, we were a little confused how to proceed, then while researching the eSport industry the team came upon a company called EEDAR. This is what their website claims:

EEDAR was founded with a belief that understanding the “DNA” of a video game through game classification would enable gaming professionals to make better business decisions. EEDAR went on to create what has become the industry’s most accurate and robust collection of video game attribute metadata.

- EEDAR

EEDAR is an eSport/Video Game analytics focused company. After reviewing a sample of their work, <https://twvideo01.ubm-us.net/o1/vault/gdc2017/Presentations/Zatkin_Geoffrey_Awesome.pdf>, the team realized that with this data set the team could act as consultants to an eSport Organization/Industry.

Business Questions

As consultants, the team would like to help the eSport Organization/Industry in 4 different ways. They would be Recruitment, Strategy – Offensive, Strategy – Defensive, and Club Management. The reason why we would like to focus on recruitment is because we need to know what style of players we need as well as what champions those players use to help with winning. The reason why we would like to focus on offensive strategy is because we want to know what champions we need to use, what objectives we get, and what spells we need to do to win. The reason why we would like to focus on defensive strategy is because we want to know what to counter each champions what strategy will be good against the team we’re playing against. The last area we would like to focus on is Club Management this is because we need to be able to quickly respond after a win/defeat and to be able to keep the talent we have by winning the most. The below questions are the questions the team developed to solve the aforementioned areas this data will help us provide business responses to an eSport Organization/Industry

* Are champ columns random or are they associated with lane assignments in League of Legends?
* Who is the champion with the highest win rate? Lowest win rate?
* What does the highest win rate champion lose to the most?
* What are the top two summoner spells for the champion with the most wins?
* What is the best composition of champions?
* What is the most banned champion? What is their win rate?
* Which objective kill count is the best predictor of winner?
* Which side has a higher win rate?
* Which first to indicator is the best predictor of winner? Which champion is present in games with the highest percentage of that indicator?
* Which champion (or team composition) is in games with the most objectives?
* What is the summoner spell with the highest win rate besides flash? Summoner spell combination?
* Which side gets more objectives in matches?
* How much is Rift Herald an indicator of win chance?
* Create a function that takes a champion name as an input and returns a win rate for that champion and the top champion they lose to.
* Create a function that takes a champion name as an input and returns the summoner spell combination with the highest win rate.
* Create a function that takes a champion name as an input and returns the champion they win the most with.

Data Analysis with Visualization & Interpretation

Answering Question #1

Business Question, Are champ columns random or are they associated with lane assignments in League of Legends?

This is a domain-knowledge question. After investigation into several rows of data, it became clear that, based on the champions chosen and their relative position within the data frame, or other key indicators such as choosing the summoner spell “Smite,” the data is organized in a random fashion. ChampID 1 does not correspond to any relative metric that would help us make further assessments of the data.

Answering Question #2

Business Question, Who is the champion with the highest win rate? Lowest win rate?

The champion with the highest win rate is Janna and the champion with the lowest win rate is Ryze. We could suggest to Riot Games to adjust the mechanics of the champion to make other champions more viable. By comparing the win rates of the top five champions, the win rates are within 2% each other. Since there is such a small difference it would not be practical to adjust Janna because of her high win rate. Janna is in line with the other viable champions. The bottom five champions are within 5% of each other. This would also not be practical to increase the viability of Ryze because of the insignificant difference. Janna has a win rate of 55% and Ryze has a win rate of 40%. An argument can be made to increase the viability of Ryze to make their win rate in line with +/- 5% of the average we see amongst the other champions.

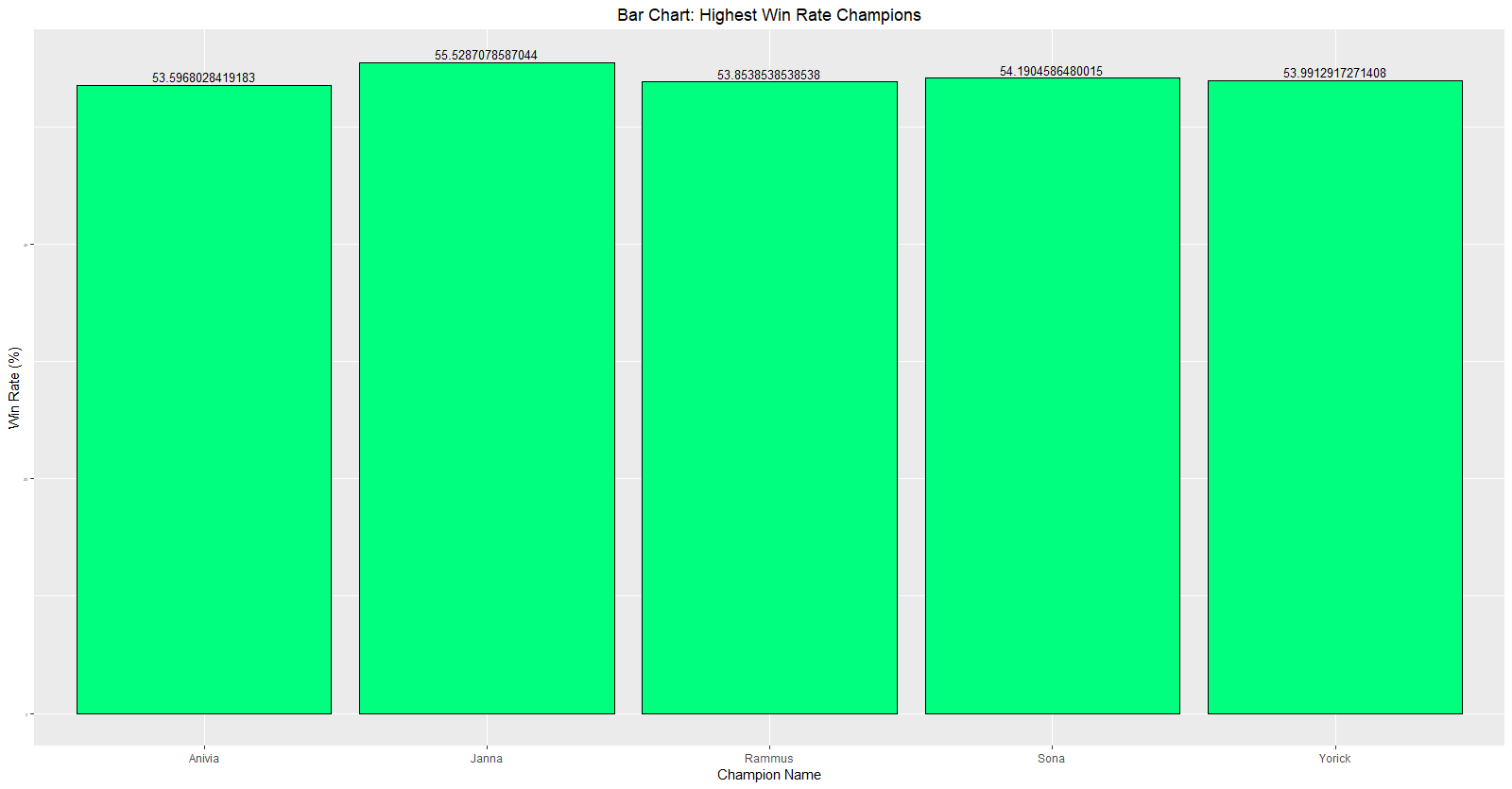


Figure 4 Top Five Champions ↑

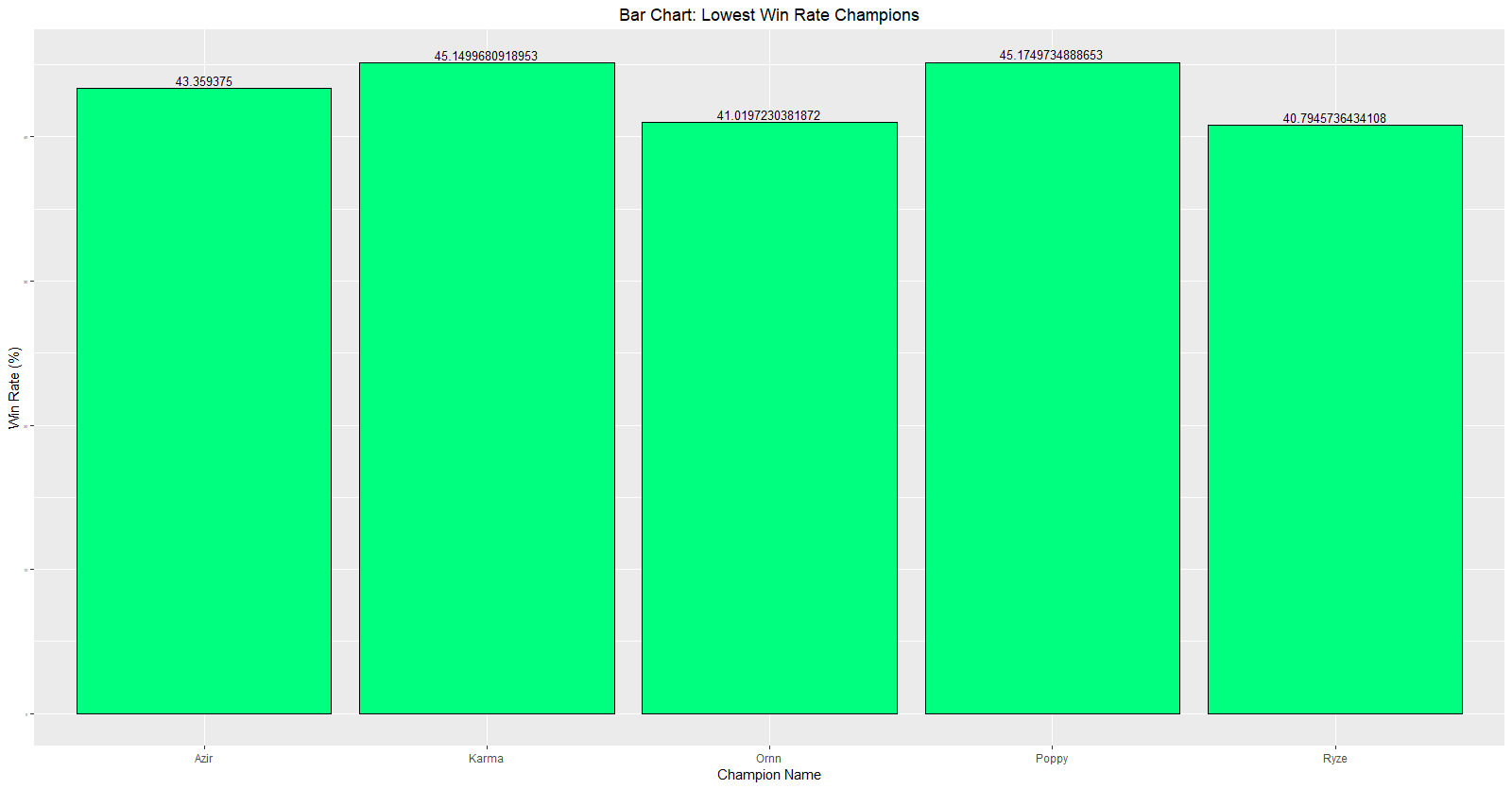


Figure 5 Bottom Five Champions ↑

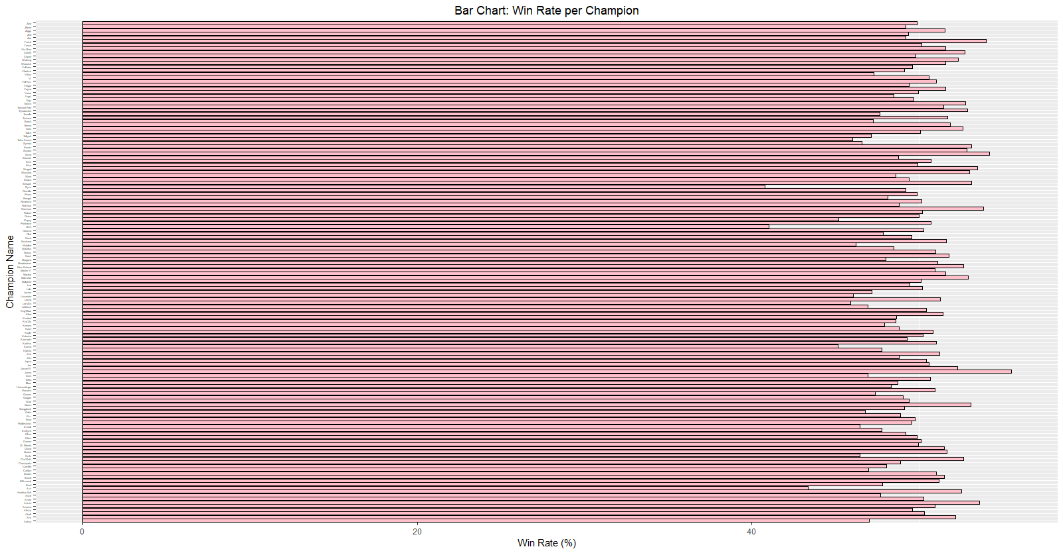


Figure 6 Win Rate per Champion ↑

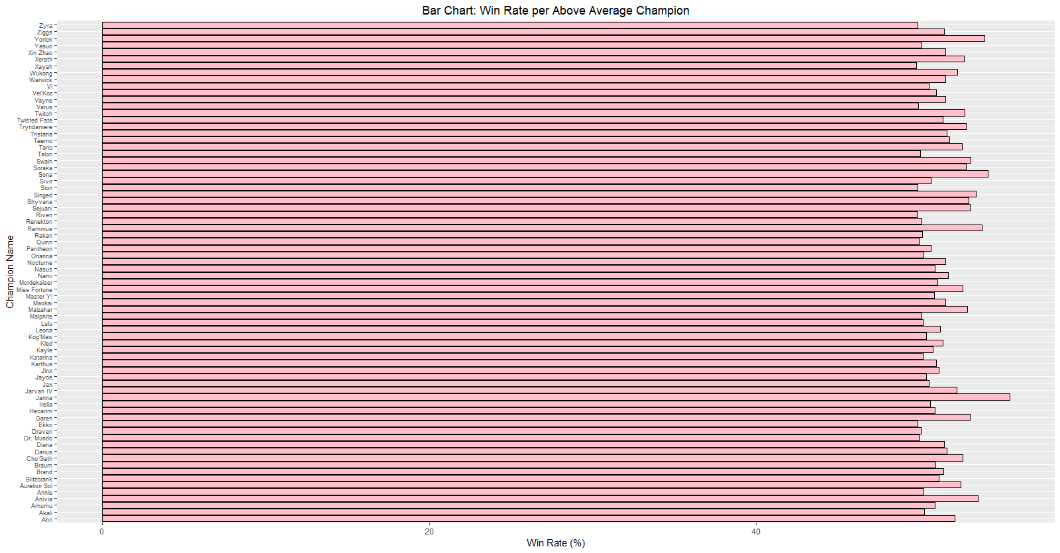


Figure 7 Win Rate per Above Average Champion ↑

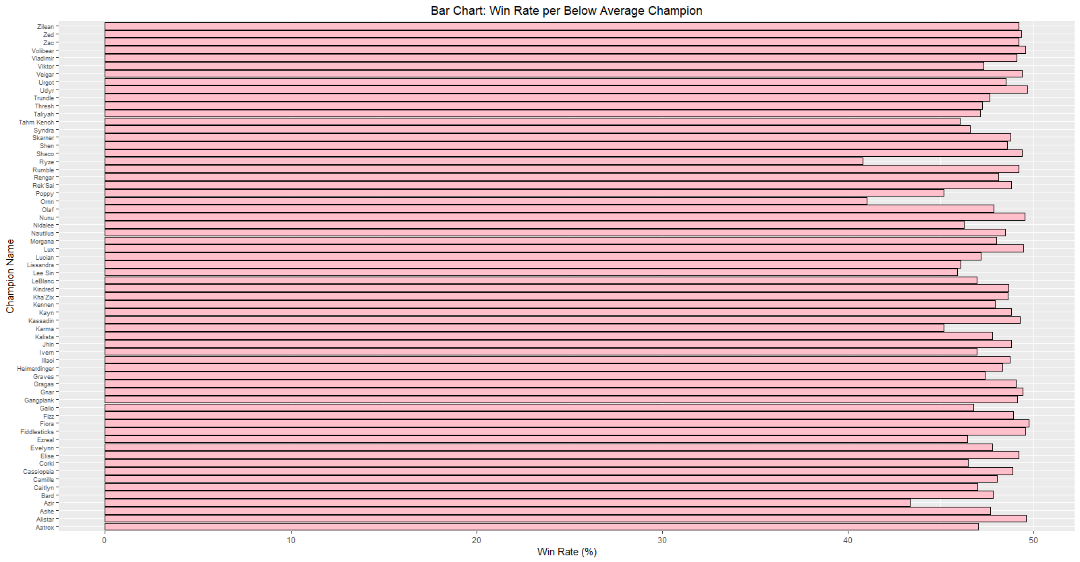


Figure 8 Win Rate per Below Average Champion ↑

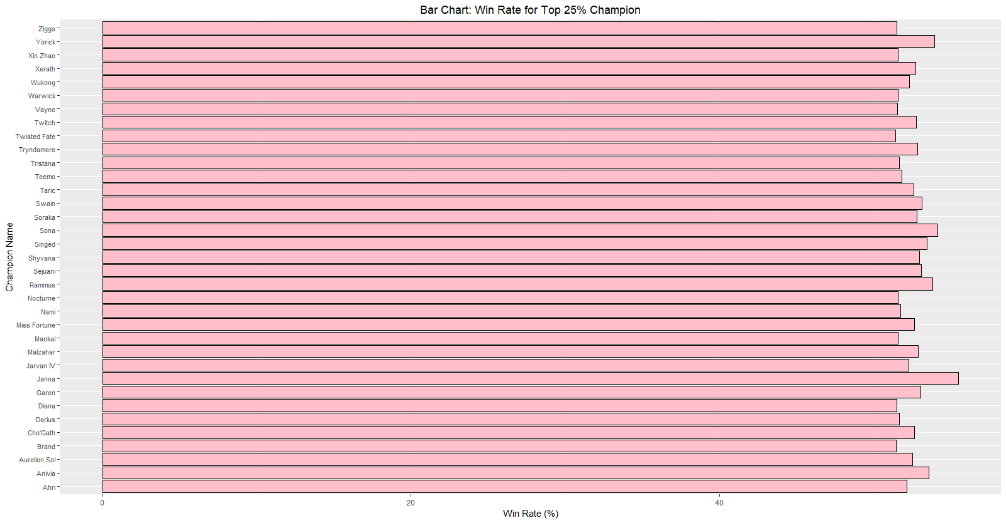


Figure 9 Win Rate for the Top 25% Champions ↑

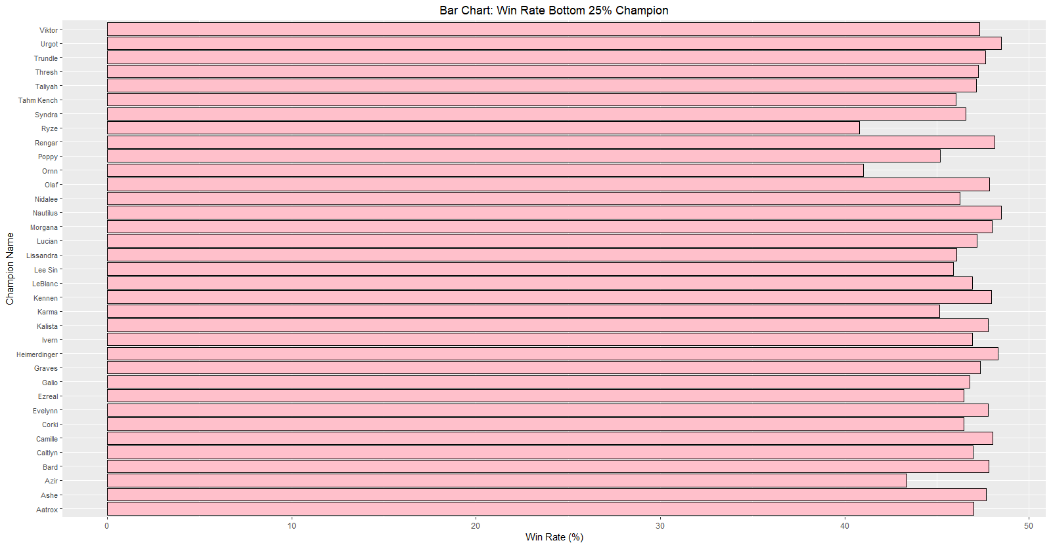


Figure 10 Win Rate for the Bottom 25% Champions ↑

Answering Question #3

Business Question, What does the highest win rate champion lose to the most?

Janna is the champion with the highest win rate. All the champions that beat Janna’s team are summed up. Thresh is the best counter to fight against Janna from Team 2. Tristana is the best counter to fight against Janna from Team 1. Overall, Thresh is the best counter for Janna.

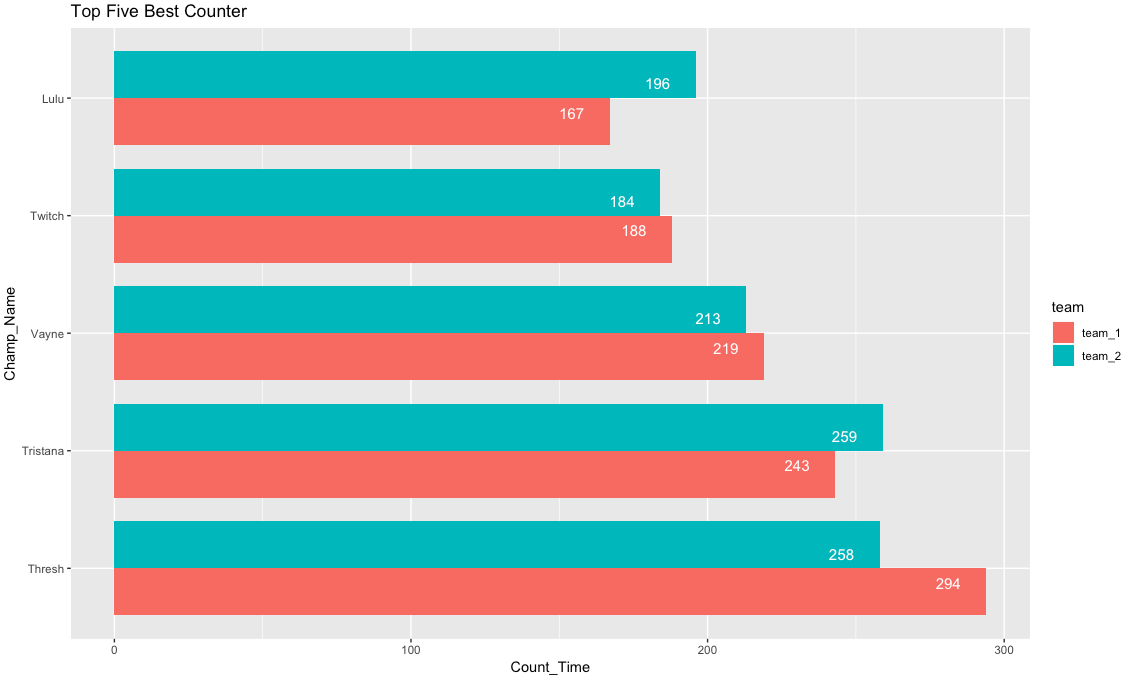


Figure 11 Top Champions that beat Janna ↑

Answering Question #4

Business Question, What are the top two summoner spells for the champion with the most wins?

After creating a vector that summed up summoner slots 1 and 2, we concluded that slot 1 most commonly contained the summoner spell Flash and slot 2 contained summoner spell Exhaust. This is useful, but summoner spell slot 1 and 2 are relative positions on the keyboard or selection preference, so the important feature is the summoner spell itself. The Team decided to combine the spells into one large vector and find the frequencies.

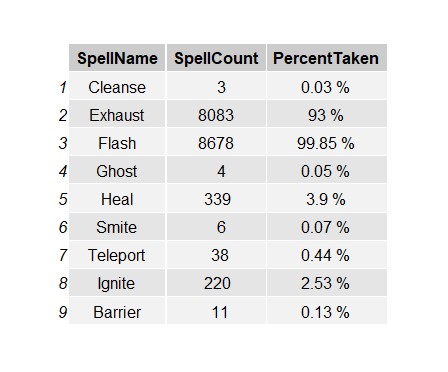


Figure 12 Summoner Spell Preference for Janna ↑

Based on the role this champion plays, these are not surprising findings. Perhaps Riot will be interested in tuning the spell Exhaust so that other spells that would be suitable for the role such as Ignite or Heal seem more appealing, or at least situationally viable. Then again, the summoner spell could be a result of the play style that is rewarded when using this champion.

Answering Question #5

Business Question, What is the best composition of champions?

There are 66 teams with the same frequency of 3 to be the winning team. The list of these teams is showed in the appendix. The following bar chart illustrates the top ten popular champions among the top 66 teams.

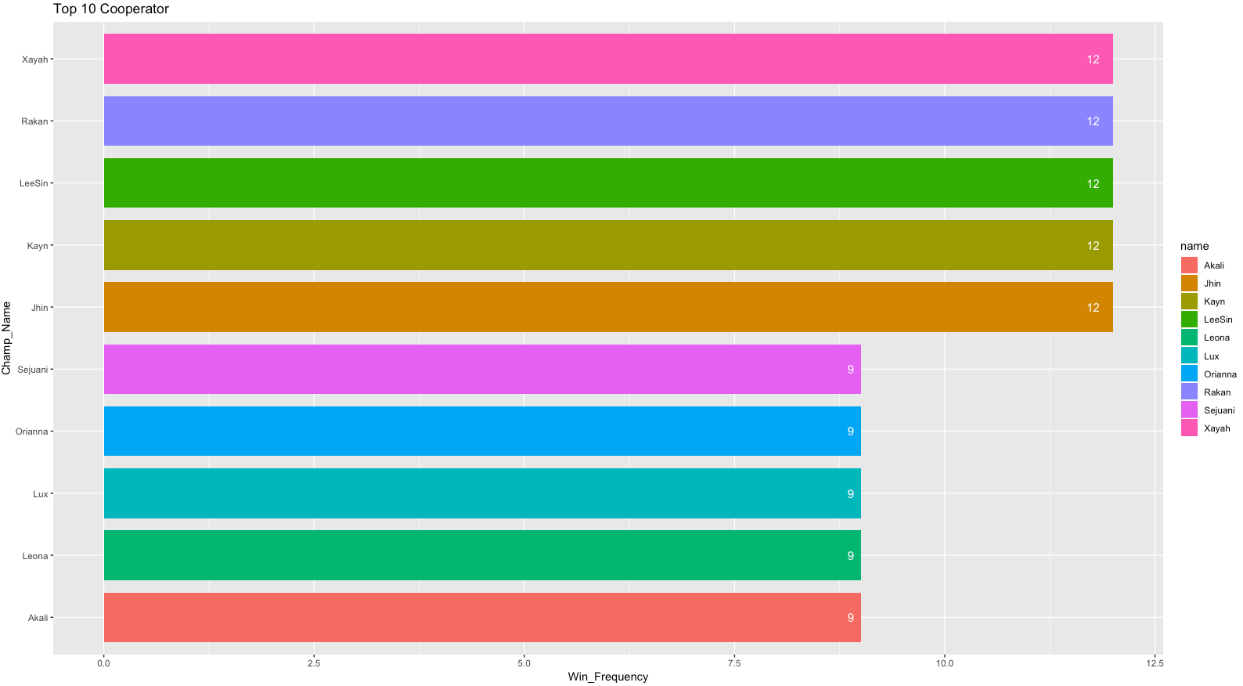


Figure 13 Bar Chart for Top 10 Most Popular Champions ↑

Answering Question #6

Business Question, What is the most banned champion? What is their win rate?

To solve this question, understanding the business domain is essential. First, all members of both teams can ban any champion of their choice, or ban no champion, so long as it is not also banned by a teammate. Second, both teams can ban the same champion. We wished to figure out the champion that players thought was the most problematic. We combined all the banned champions into one vector and found the most common occurrence, which was Yasuo. Yasuo has a win rate of 50.15% (Mean Win Rate = 49.77%), which is high in League of Legends. The ban rate for Yasuo totaled 64.12%, which is much higher than any of the other champions in the data set.

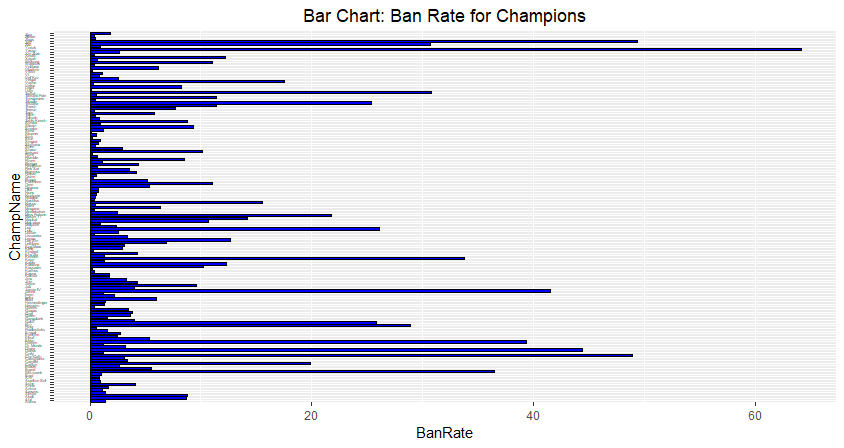


Figure 14 Bar Chart: Ban Rate for Champions ↑

The ban rate for champions varies wildly and is usually based on the perceived strength of a champion. For League of Legends players, Yasuo is considered the staple frustrating pick to fight against given his champion ability kit. Looking at his win rate will help us understand how strong he is given games he participates in. We can also look at the top 10 champion ban rates along with their win rates to determine how appropriate these bans are.

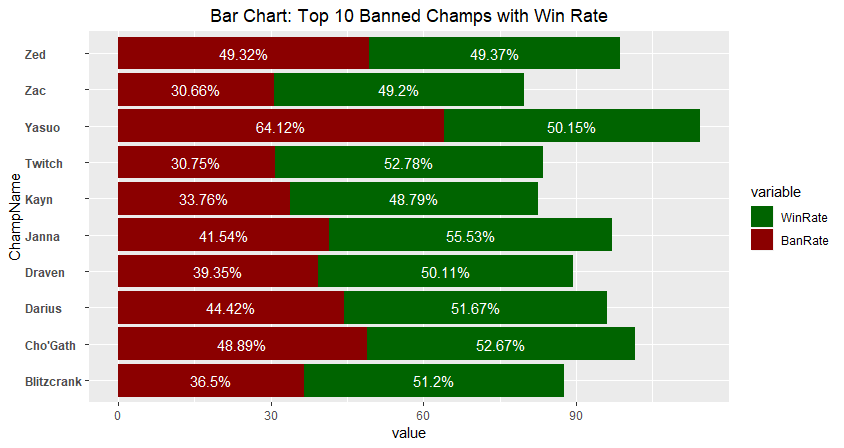


Figure 15 Bar Chart: Top 10 Banned Chaps with Win Rates ↑

Of the top ten banned champions, only 3 of them have a win rate below average (Zed, Zac, and Kayn). Yasuo has an above average win rate, and the champion with the highest win rate, Janna, is also in the top ten banned champions. Based on the respective win rates, it appears the top ten banned champions have various qualities that make them difficult for other players to deal with reliably. Using this visualization, we can inspect possible routes for balancing champions or items, so these champions do not feel so necessary to ban for players.

Answering Question #7

Business Question, Which objective kill count is the best predictor of winner?

There is a total of 5 different objective types that each team can take. All these objectives vary in the number available each game and the frequency with which they can be taken. As such, it is important to understand which objectives are important and determine if there is an objective that leads to more victories than another. A simple way to do this is to scale each column of data so that they may all be weighted equally in comparison with each other. A quick plot of the winning team against the blue side tower kills will help us understand the data better.

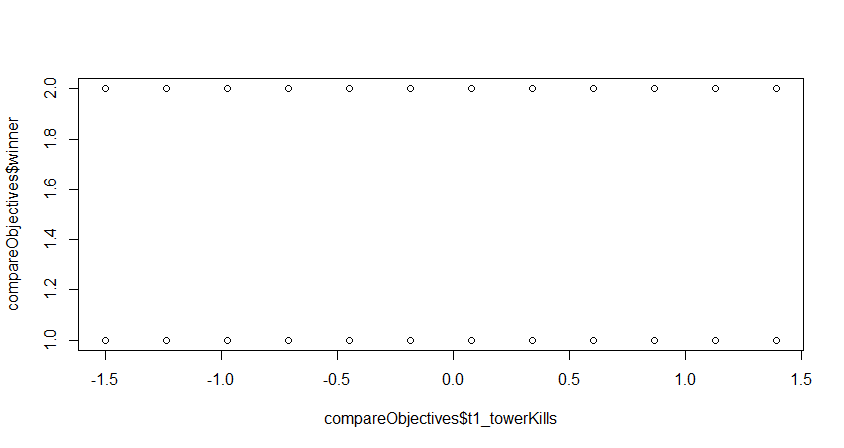


Figure 16 Game Winner by Team 1 Tower Kills ↑

Given the nature of the data, it is not advisable to model this relationship linearly. Therefore, The Team implemented a logit model for assessing binary data. The winning team values were changed: Blue winning became 0 and Red became 1. A logit analysis was then performed using all the variables.

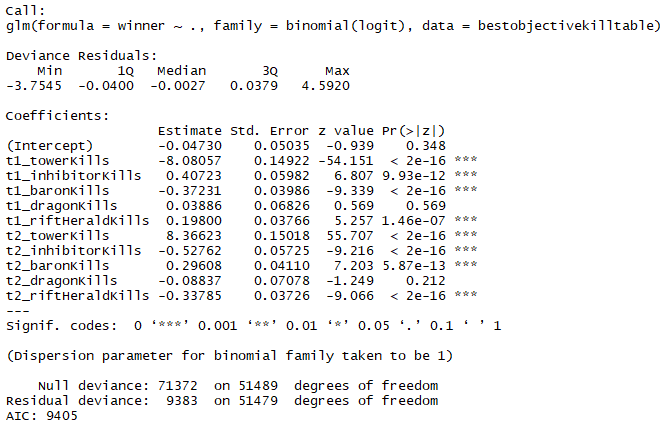


Figure 17 Generalized Linear Model Summary ↑

Based on this data, all variables are significant except for dragon kills. Next, we performed a variance inflation factors test.

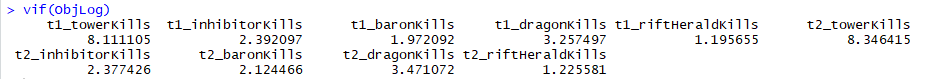


Figure 18 VIF Test to test for collinearity ↑

There is a semblance of multicollinearity between the tower kills for t1 and t2. Because the collinearity is not greater than 10, the tower kills variables was kept as is and the dragon kills variables were eliminated. The logit analysis was then run again after eliminating the t2\_towerkills variable from the data set.

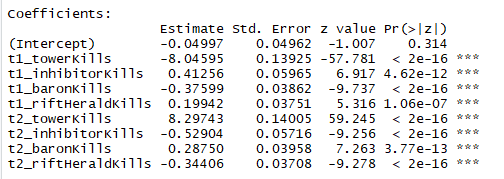


Figure 19 Logit Analysis for Prediction ↑

The model is then called upon to predict the winner of a match when given these variables. The model was successful with prediction 95.97% of the time.

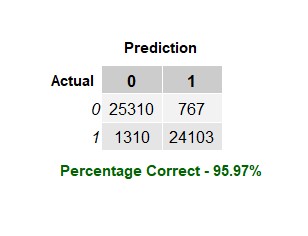


Figure 20 Logit Model Confusion Matrix to Calculate Accuracy ↑

Using just t1\_towerkills as our x variable again, we can plot the Logit model to understand the circumstances under which one team has a higher probability to win than another.

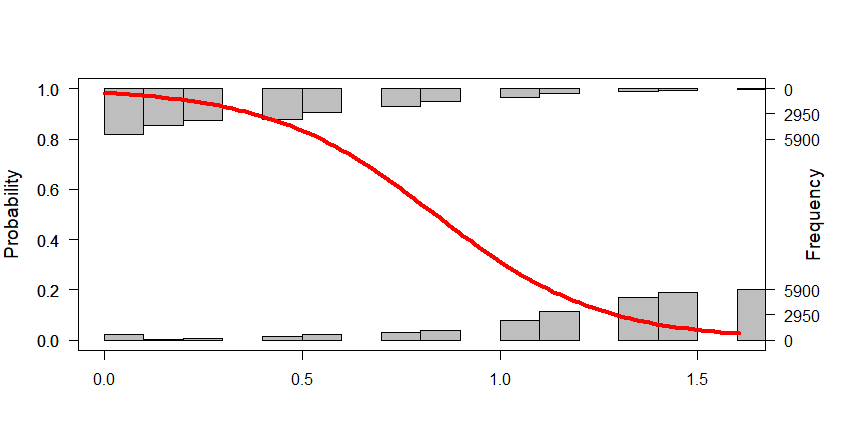


Figure 21 Probability of Winning Team as Demonstrated by t1\_towerkills ↑

To conclude, tower kills by either side is the most significant measure of the winner of the game. Because all these variables have been scaled, the coefficient is the only indicator we need to understand how powerful of an indicator this variable is. Given the number of towers per team per game (each side has 11 turrets and 3 inhibitors), it is understandable why this factor is so powerful. Towers must be taken in order to win. Most of the other objectives – Baron, Dragon, and Rift Herald – do not need to ever be secured in order to win but are there to add factors to securing a victory. Possible design areas to address given this information would be tuning dragons so they are either easier to get or give a greater reward when killed. Others would be to increase the potency of Baron Nashor and Rift Harold or introduce items that benefit from those objectives directly. Either way the effect of such a prominent objective on the map should not go unaddressed.

Answering Question #8

Business Question, Which side has a higher win rate?

Team 1 in blue has a higher win percentage. However, the gap between the two sides is not big. The game seems to be fair enough, with no side having a definite advantage over the other one.

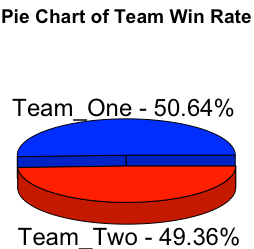


Figure 22 Pie Chart of both Team Win Rate ↑

Answering Question #9

Business Question, Which first to indicator is the best predictor of winner? Which champion is present in games with the highest percentage of that indicator?

The team that scores the first objective in a match is usually the team that can get an early advantage. League of Legends is designed in such a way that the team that takes advantage of these early rewards will be the one that prevails in the end. According to the data, this is usually the case. Based on linear regression, the first team to score an inhibitor, followed by the first tower, will have the greatest chance of winning. Intuitively, the point in which a team can destroy an inhibitor usually indicates that the team in the lead has pressed their advantage hard enough that they are on the brink of victory. (Rift Herald was removed, p > 0.05.)

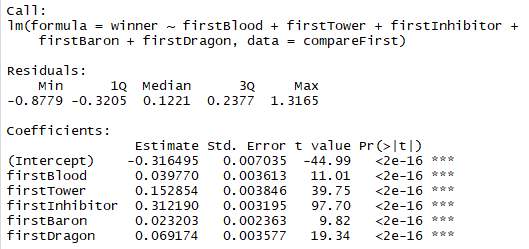


Figure 23 Linear Model Summary of First Kill Objectives ↑

This data about the most important factor is right in line with what Riot expects to see regarding the trend in gameplay, but Rift Herald may be an objective that needs to be retuned to have an even greater impact on deciding the winner.

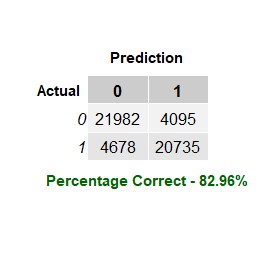


Figure 24 Prediction using Linear Model and First Kill Objectives ↑

Answering Question #10

Business Question, Which champion (or team composition) is in games with the most objectives?

In Team 1, the following team had the most objectives

* Champion Slot 1: Taric
* Champion Slot 2: Katarina
* Champion Slot 3: Lee Sin
* Champion Slot 4: Xayah
* Champion Slot 5: Kayle

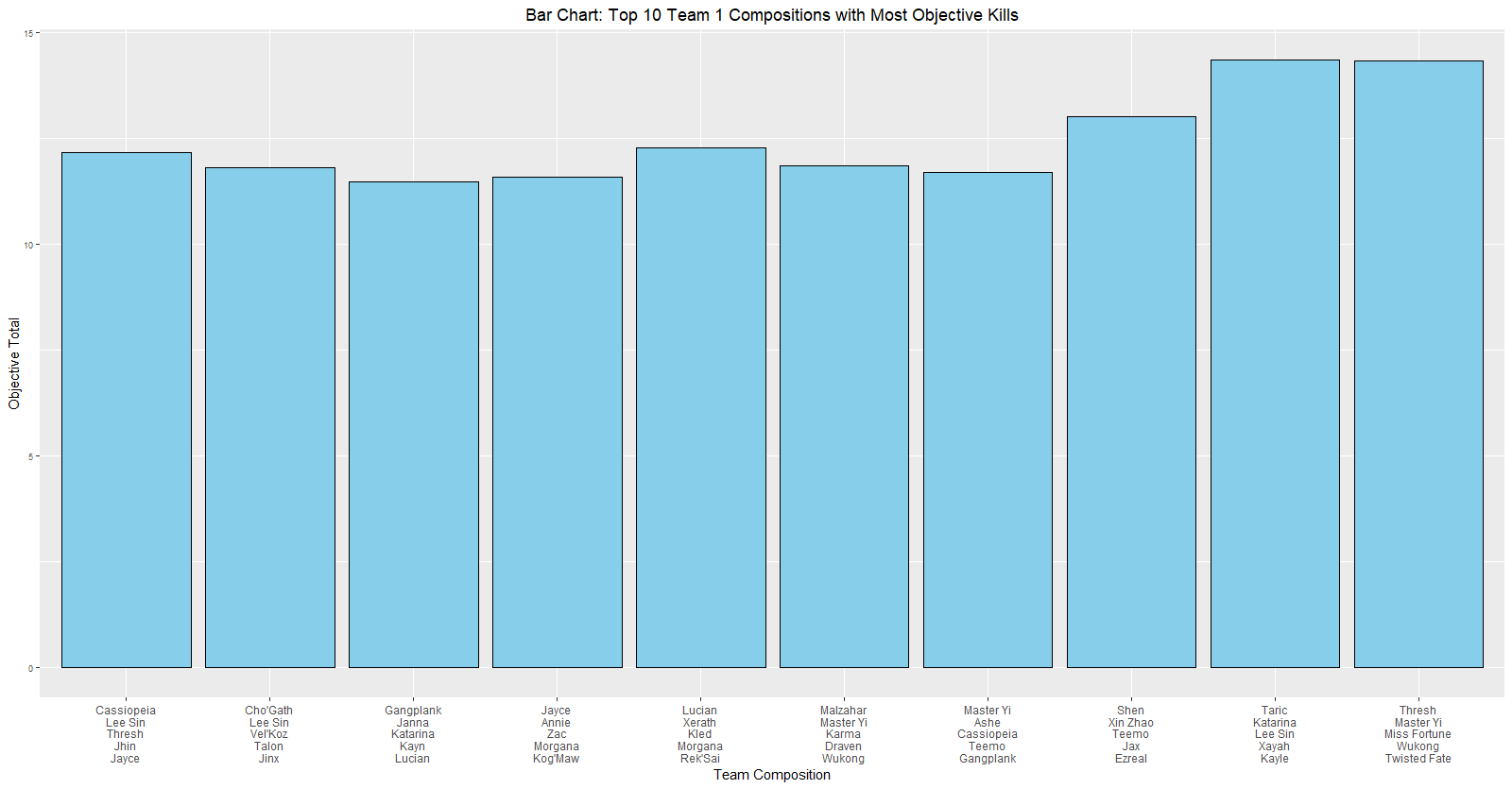


Figure 25 Top 10 Team 1 Compositions with the Most Objective Kills ↑

In Team 2, the following team had the most objectives

* Champion Slot 1: Lucian
* Champion Slot 2: Master Yi
* Champion Slot 3: Twitch
* Champion Slot 4: Morgana
* Champion Slot 5: Trundle

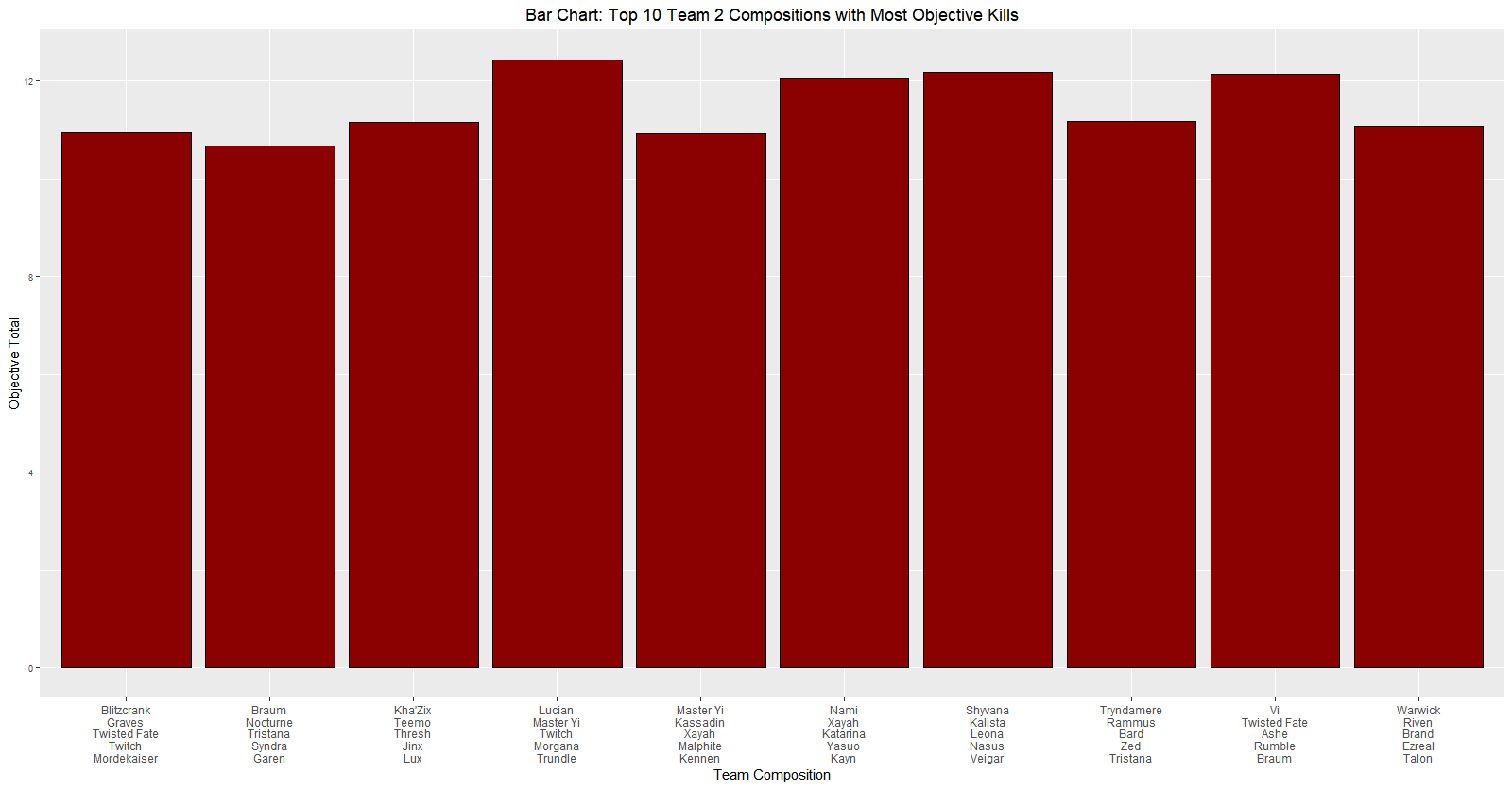


Figure 26 Top 10 Team 2 Compositions with the Most Objective Kills ↑

Answering Question #11

Business Question, What is the summoner spell with the highest win rate besides flash? Summoner spell combination?

First it is important to understand the distribution of the data from a player perspective.



Figure 27 Total Summoner Spell Frequency Choice of All Players ↑

Flash is a staple spell choice and all of League of Legends gameplay is designed with this ability in mind. It is not surprising to find that most players choose this spell. Because of how common it is, we would not be surprised that Flash had a 100% win rate and a 100% loss rate due to the likelihood of the spell being chosen by a member of both teams. The same can be said about smite, as each jungler is expected to take the rune. It would be more useful to observe the data from a champion or role perspective to understand how each spell affects the outcome of a match and revise our business question. Since Yasuo is the highest banned champion, we observed his win rate based on summoner spell.

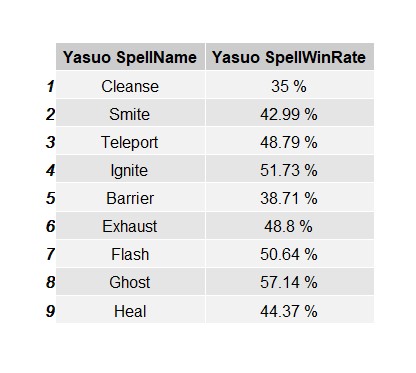


Figure 28 Yasuo Win Rate Based on Summoner Spell ↑

Yasuo’s total win rate is 50.15%. Based on this, we can understand what pattern of play for this champion is rewarded. Cleanse, Barrier, and Heal are defensive summoners and a player that takes these spells on Yasuo is more likely to lose than someone who takes more aggressive spells such as Ignite or Ghost. We can also observe Janna, the champion with the highest win rate.

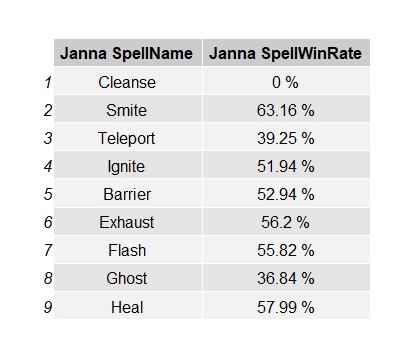


Figure 29 Janna Win Rate Based on Summoner Spell ↑

Janna’s win rate is 55.53%. Here we see supportive style of play is most rewarded, with Heal and Exhaust granting more wins than aggressive play, such as Ignite and Ghost. Additionally, we can investigate why Janna, a support class champion, has such a high win rate when taking Smite, a champion that junglers take. Also, with this information, Riot can understand how champions perform given the bounty of summoner spells and move to tune spells or champions based on their win rates given a spell.

Answering Question #12

Business Question, Which side gets more objectives in matches?

Team 2 gets more objectives. We would need more information about which game map these games took place to explain this. The maps may be asymmetric and favor Team 2’s side over Team 1. This is interesting because Team 1 has a higher win rate even though they take less objectives on average. When we observe the actual values as per the figure below, on average Team 2 gets slightly more objectives than Team 1. This difference may be insignificant.

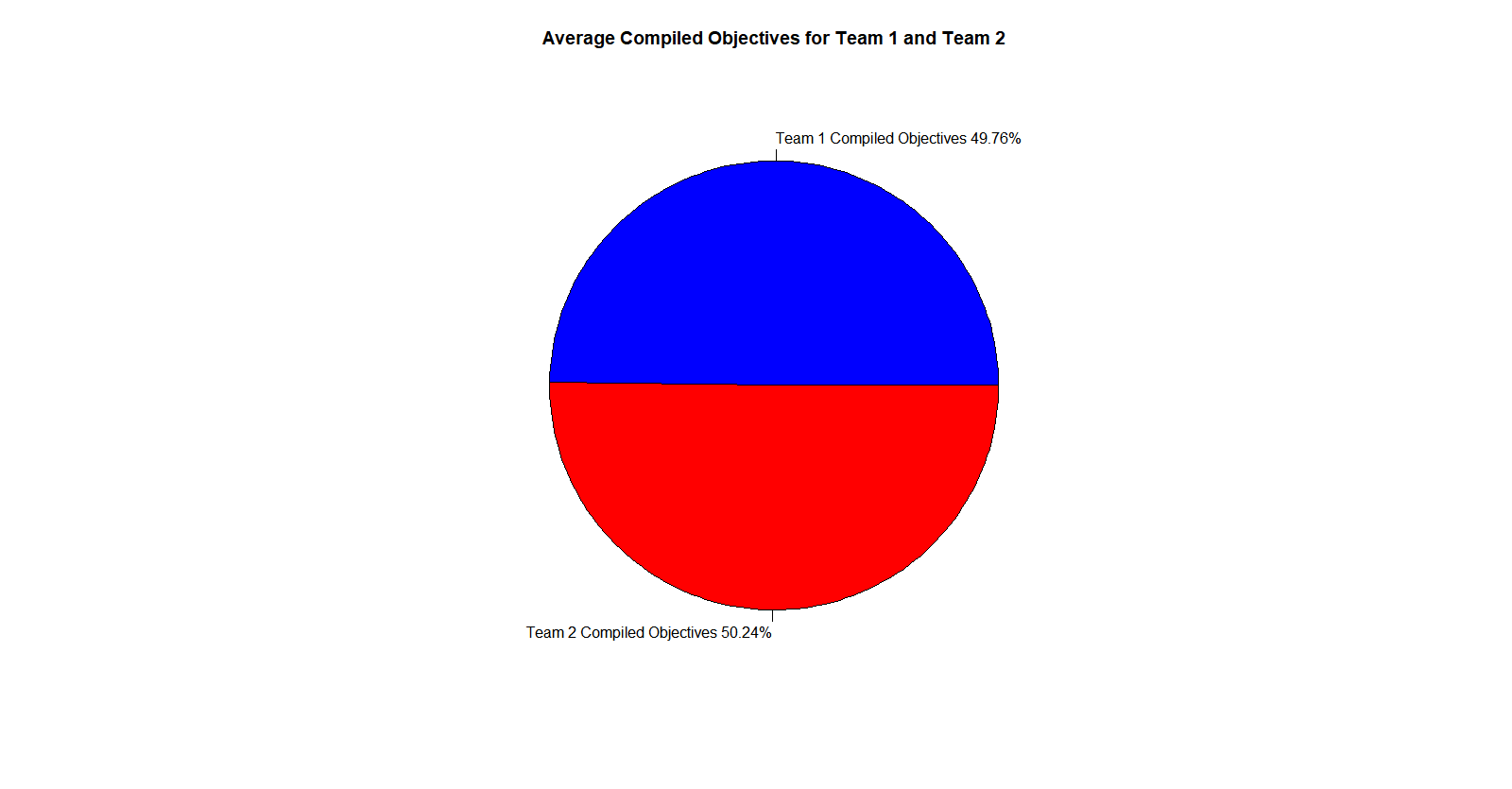


Figure 30 Team 1 Compiled Objectives vs Average Team 2 Compiled Objectives ↑

Can we predict the outcome of a game based on the total objectives of both team 1 and team 2? Please note two caveats with the plots. First, a random subset of data was taken to generate this plot due to computational constraints (7500 out of 50000 cases). The model may not be truly representative of the population. Second, Team 1 is referred to 1 and Team 2 is referred to 0. Both kvsm and naïve Bayes models can predict the outcome of a match based on the compiled objectives of both teams with at least 91% accuracy. However, since the model is based on a subset of the original dataset, we cannot be certain that the model can be generalized to the larger population dataset. Our model has difficulty discerning the outcome of a match when both teams have an equal number of objectives. This is another deficiency with our model.

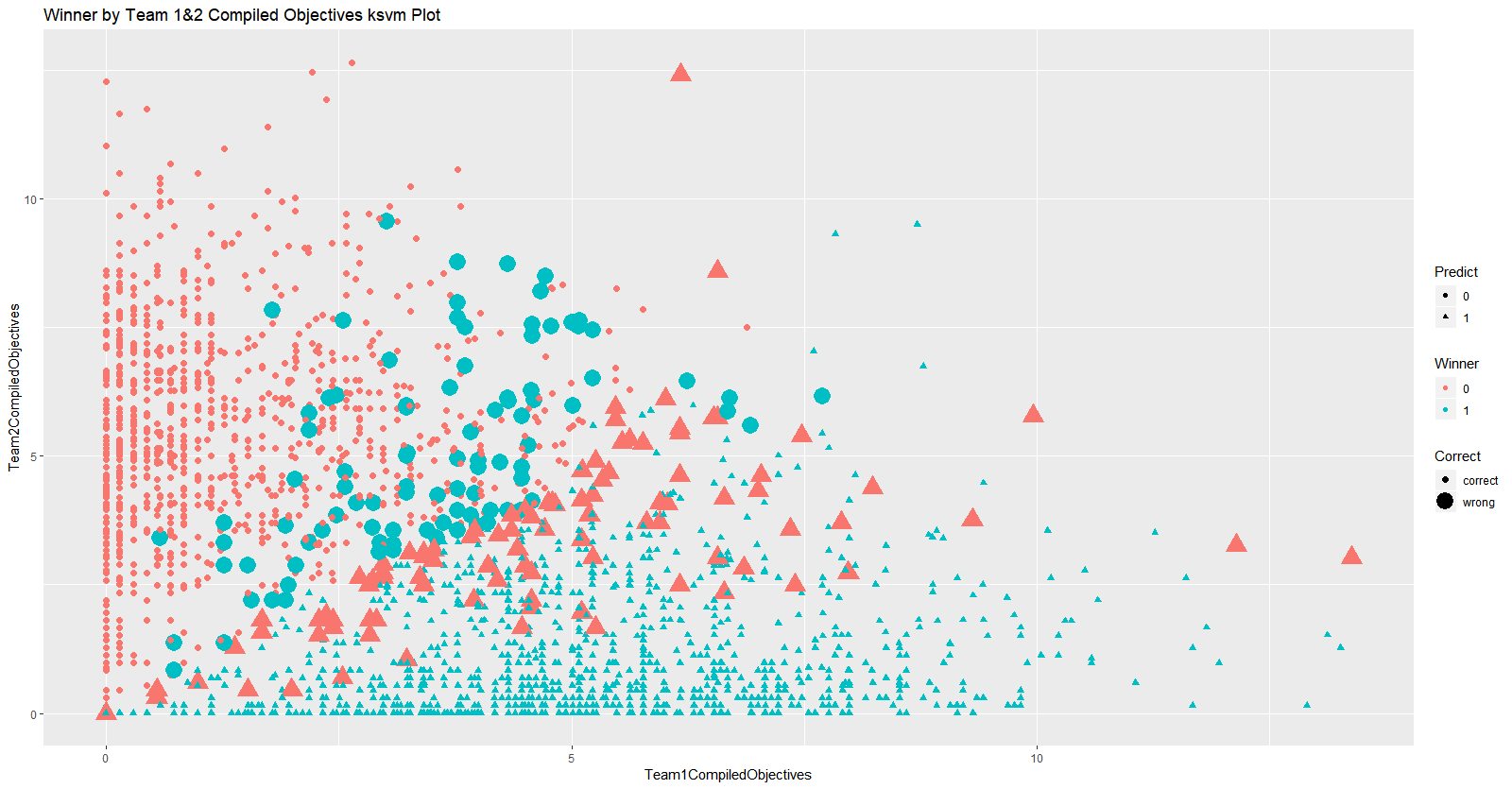


Figure 31 ksvm Plot – Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑

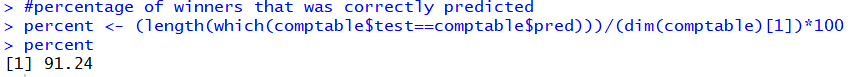
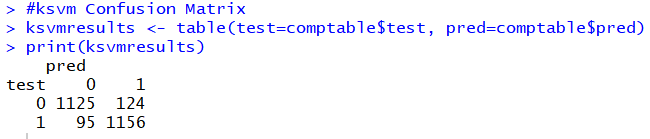


Figure 32 ksvm Confusion Matrix and Accuracy ↑

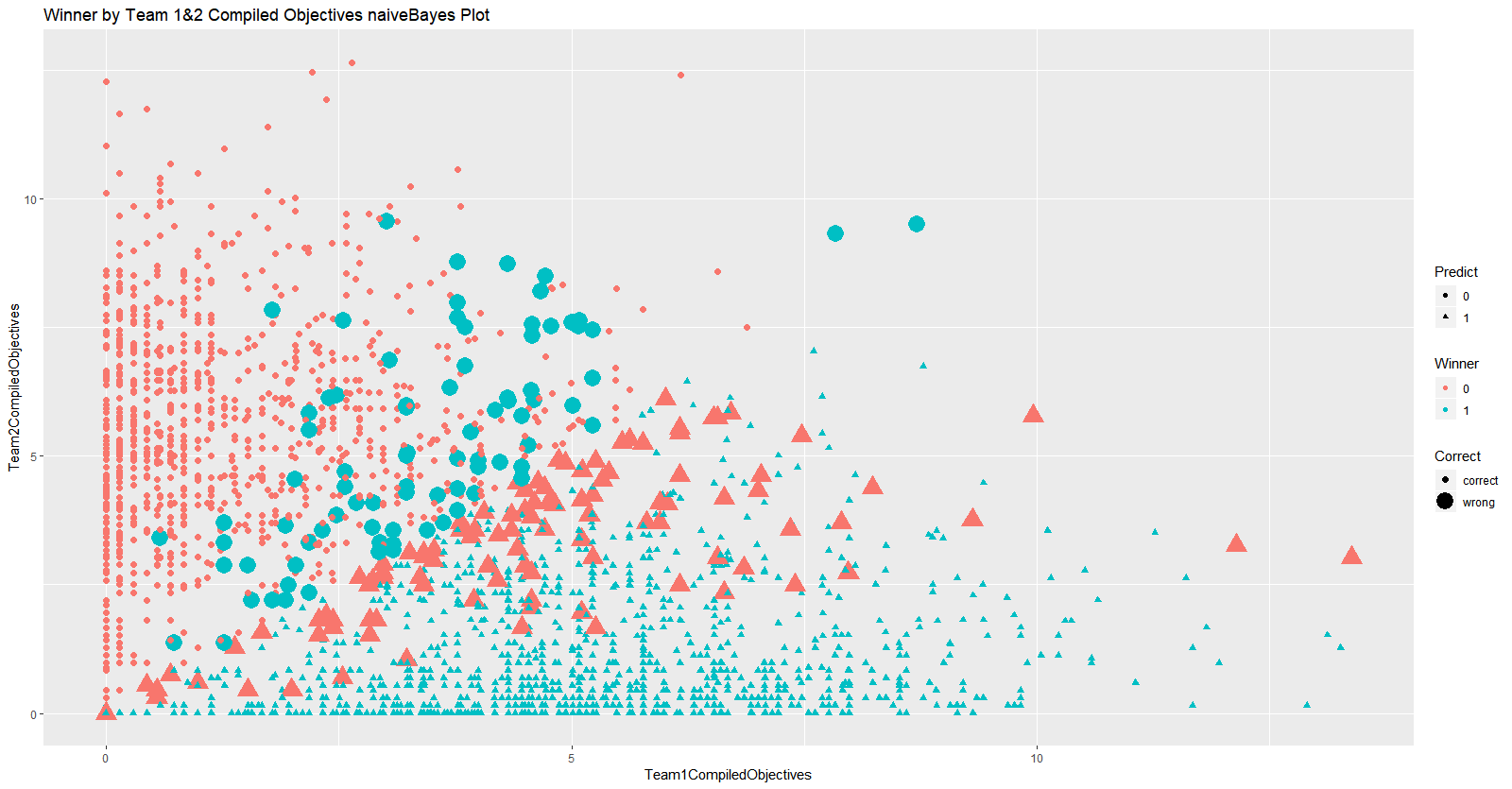


Figure 33 Naïve Bayes Plot– Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑

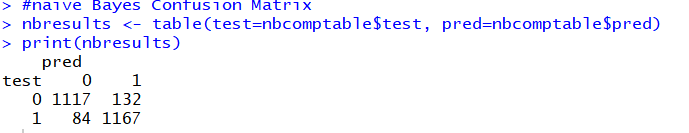
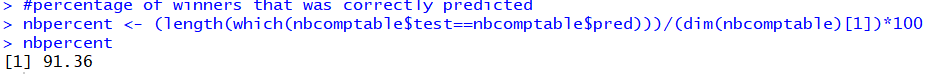


Figure 34 Naïve Bayes Confusion Matrix and Accuracy ↑

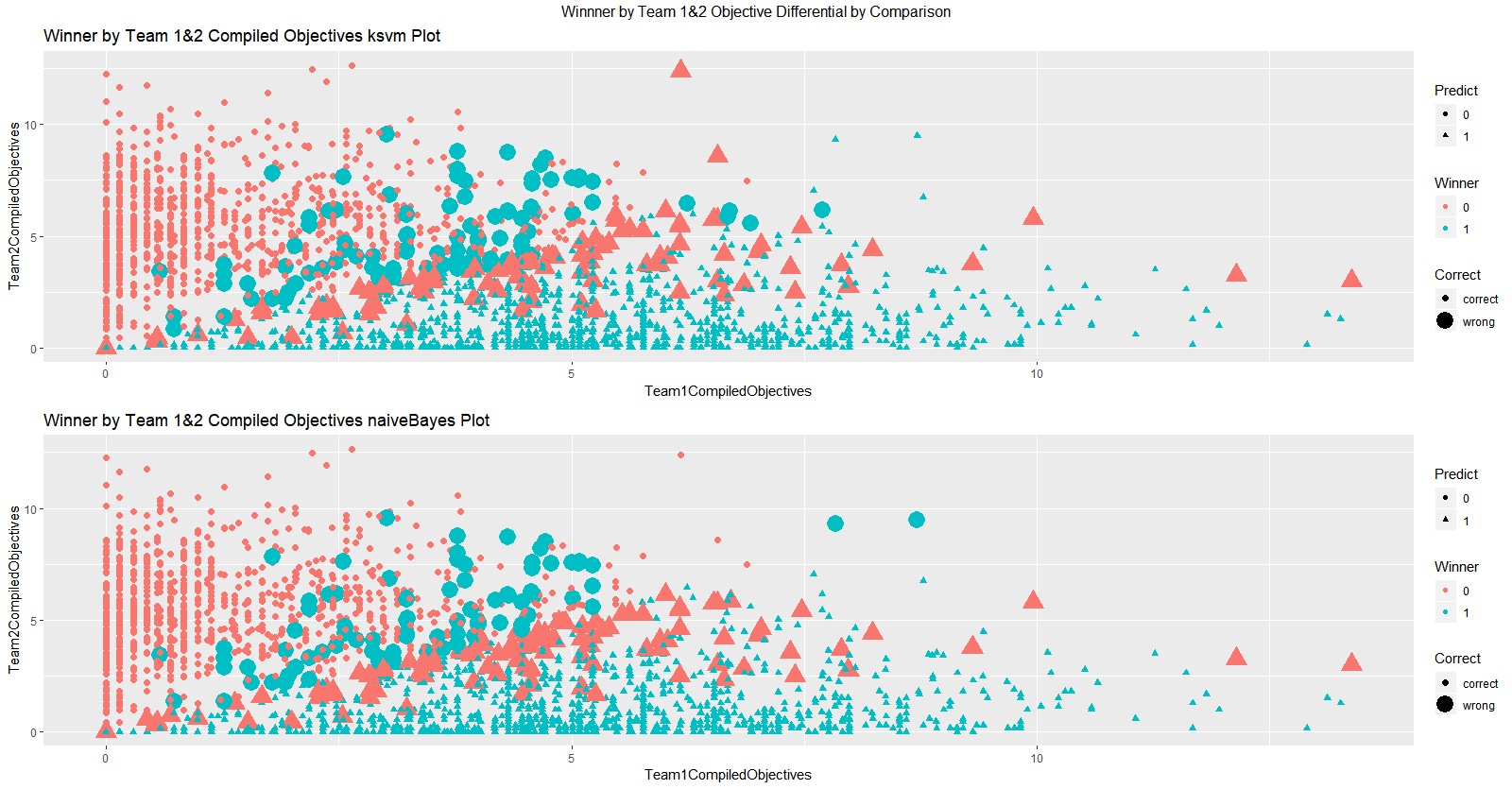


Figure 35 Model Comparison - Accuracy to predict the Winner based on Team 1 & 2 Compiled Objectives ↑

Answering Question $13

Business Question, How much is Rift Herald an indicator of win chance?

65% of wins occur without the Rift Herald kill. Only 35% of wins occurred with the Rift Herald kill. If we look at a specific team, a team wins more games without a Rift Herald kill than with a Rift Herald kill. A Rift Herald kill may not be a significant objective to take for winning a game.

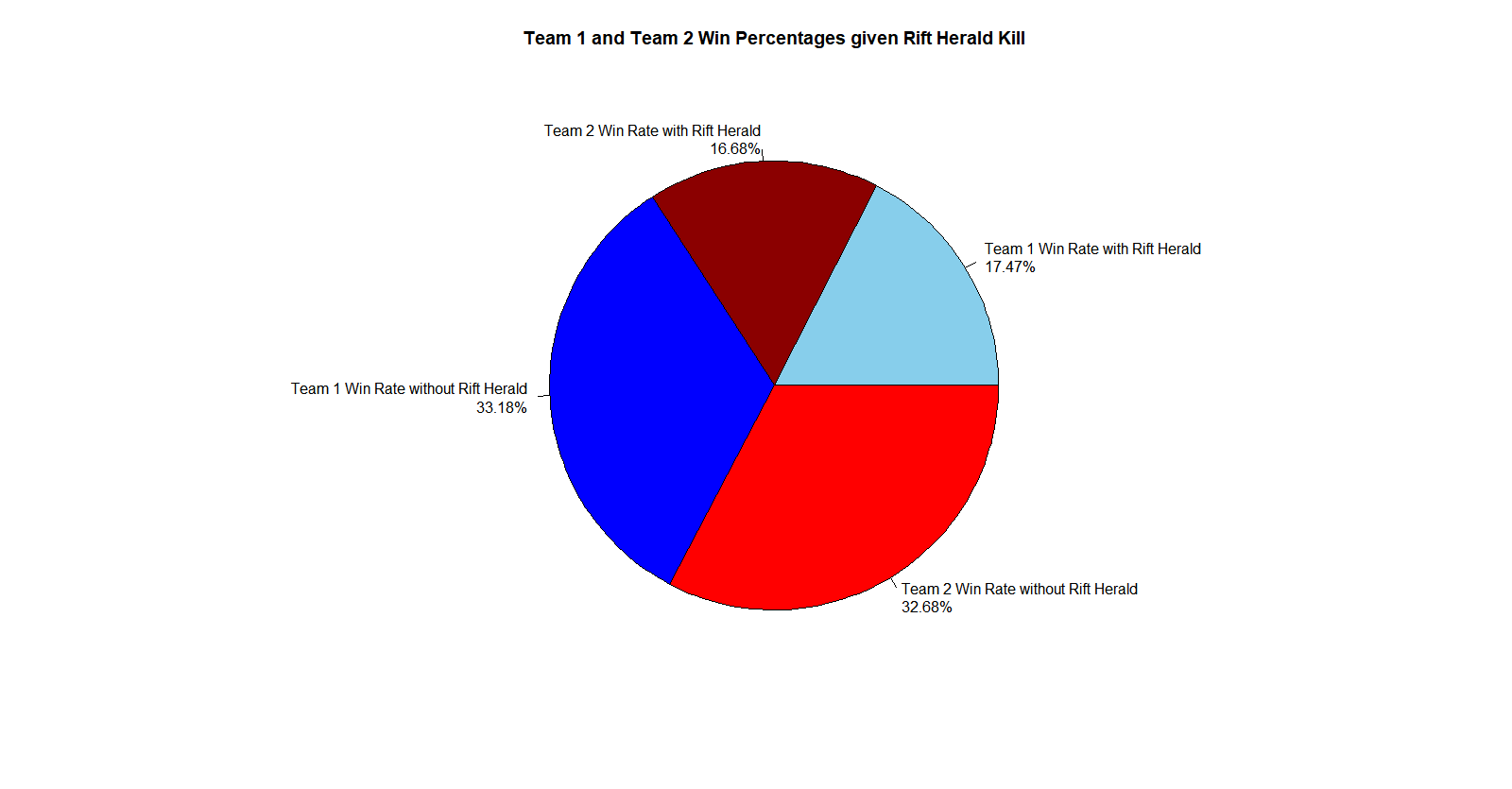


Figure 36 Team 1 and Team 2 Win Percentages given Rift Herald Kill ↑

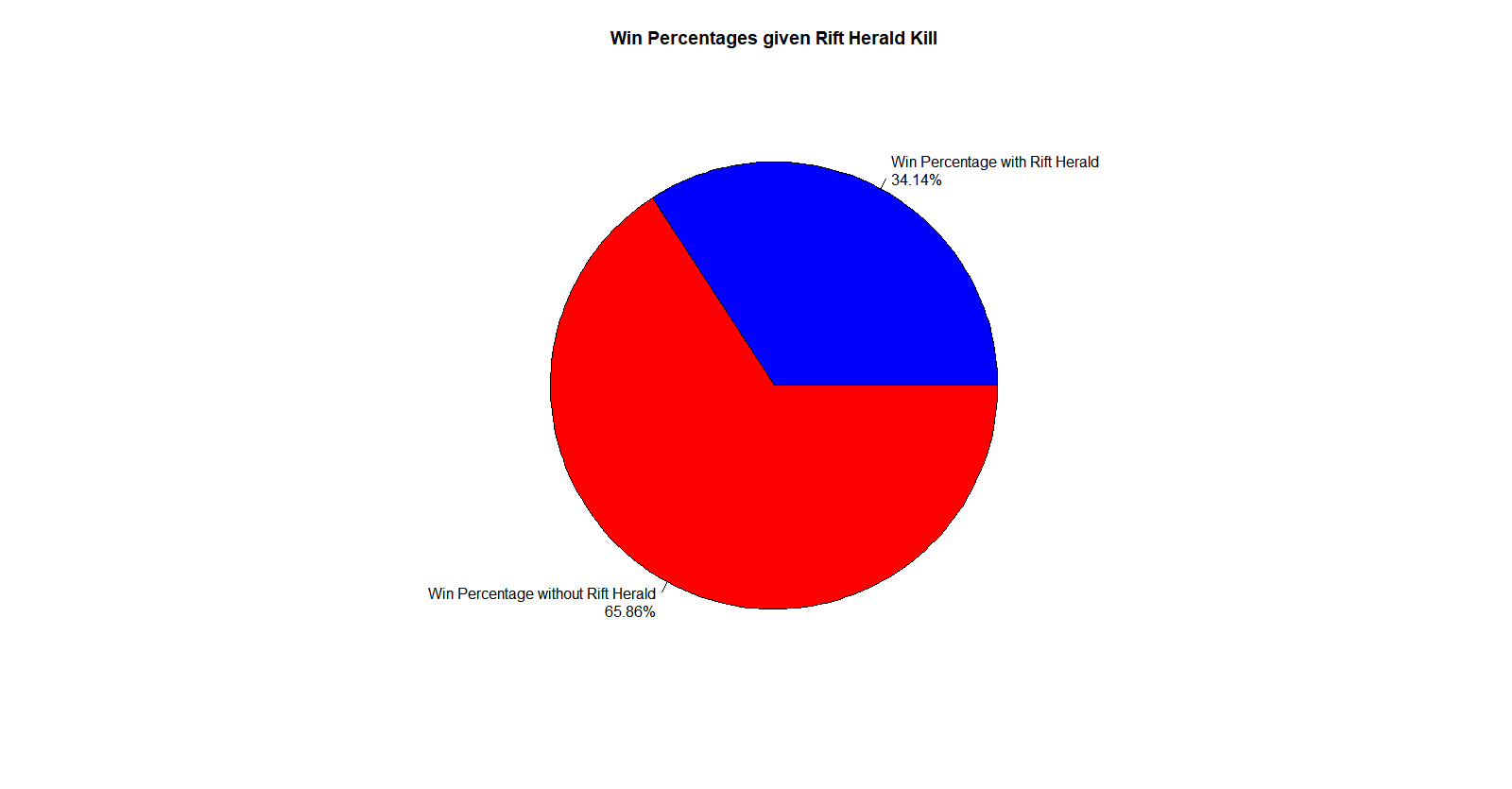


Figure 37 Win Percentages given Rift Herald Kill ↑

Answering Question #14

Business Question, Create a function that takes a champion name as an input and returns a win rate for that champion and the top champion they lose to.

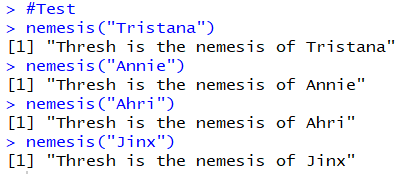


Figure 38 Testing the nemesis function ↑

Answering Question #15

Business Question, Create a function that takes a champion name as an input and returns the summoner spell combination with the highest win rate.

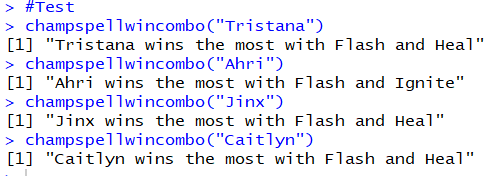


Figure 39 Testing the champspellcombo function ↑

Answering Question #16

Business Question, Create a function that takes a champion name as an input and returns the champion they win the most with.

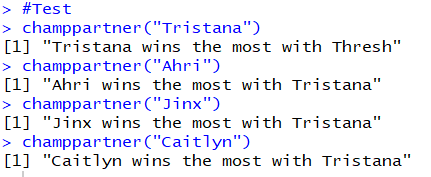


Figure 40 Testing the champpartner function ↑

Conclusions - Actions Insights

Review

After answering the above 16 questions the team gained a lot of insight into how the game operated and successful strategies to win. The team was able to answer these questions by writing and developing R code to solve this. The team developed over 2.5k lines of code. In the code there are 16 different functions which help promote automation and helps the team quickly respond since it does the development automatically. Additionally there are 3 separate prediction models 2 with 90+% accuracy and 1 with 80+% accuracy. Therefore we have several different ways of analyzing our data and estimating if our predictions are correct. All in all the team used data analytic strategies of Data mining, linear Modeling, importing in data, and writing functions to accomplish this goal of solving the above questions. In review, the below areas of Strategy, Champions, and Team answer our questions from above about Recruitment, Strategy – Offensive, Strategy – Defensive, and Club Management. Below are the highlights from those sections and below that is the R-code to how we answered those questions.

Strategy

The best combination is the champion Janna with the summoner spells Flash and Exhaust.

A tower kill is the most significant measure to determine which team will win a match.

A Rift Herald is not significant in determining an outcome of match. Potential re-work of this mechanic may be needed to increase its impact.

Adjustments need to be made to address certain champion and summoner spell combinations.

Given the objectives of both teams, an accurate prediction can be made to determine the outcome of a match.

Champions

Janna wins the most and Ryze wins the least.

Thresh is the best counter to Janna.

Yasuo is the most banned champion.

Teams

Both teams are evenly matched. The win percentages of both teams is near to a 50/50 split, and on average, both teams have the similar objective totals.

Team 1 composition with the most objectives is Taric, Katarina, Lee Sin, Xayah, and Kayle.

Team 2 composition with the most objectives is Lucian, Master Yi, Twitch, Morgana, and Trundle.

Appendix - R code

Environment configuration

# if you did not install the following packages, please do so

# if (!requireNamespace("BiocManager"))

# install.packages("BiocManager")

# BiocManager::install()

# BiocManager::available()

# use the following format to install lib for R 3.5.1

# the default install.package() function may not work for the latest R version

# BiocManager::install('genefilter')

# install.packages('plyr')

# install.packages('jsonlite')

# install.packages('genefilter')

# install.packages('modeest')

# install.packages('ggplot2')

# install.packages('plotrix')

# install.packages(‘car’)

# import library dependency

# for any business question, please import the lib here

library("plyr")

library("jsonlite")

library("modeest")

library("ggplot2")

library(plotrix)

library(car)

Import Data

# set work directory

getwd()

setwd('../data/league-of-legends/')

# set the data source

GAMES <- 'games.csv'

# CHAM\_INFO\_1 <- 'champion\_info.json'

CHAM\_INFO\_2 <- 'champion\_info\_2.json'

SUM\_SPELL <- 'summoner\_spell\_info.json'

# load csv and json

games\_df = read.csv(GAMES)

# cham\_info\_1\_list = fromJSON(CHAM\_INFO\_1) # type, version, data

champ\_info\_2\_list = fromJSON(CHAM\_INFO\_2) # type, version, data

# cham\_data\_1\_list = cham\_info\_1\_list$data # title, id, key, name

champ\_data\_2\_list = champ\_info\_2\_list$data # tag, title, id, key, name

sum\_spell\_info\_list = fromJSON(SUM\_SPELL)$data # 'id', 'summonerLevel', 'name', key': 'SummonerHeal', 'description'

# convert champ list to data frame

champ\_prep <- function(champ\_list){

champ\_df <-as.data.frame(do.call("cbind", champ\_list))

champ\_df <- data.frame(r1=names(champ\_df), t(champ\_df))

champ\_df$r1 <- NULL

row.names(champ\_df) <- NULL

colnames(champ\_df) <- c('ChampTag', 'ChampTitle', 'ChampID', 'ChampionKey', 'ChampionName')

return(champ\_df)

}

champ\_df <- champ\_prep(champ\_data\_2\_list[-1])

# convert sum spell info list to data frame

sum\_spell\_prep <- function(sum\_spell\_list){

sum\_spell\_df <- data.frame(matrix(unlist(sum\_spell\_info\_list), nrow=length(sum\_spell\_info\_list), byrow=T), stringsAsFactors = FALSE)

colnames(sum\_spell\_df) <- c("SummonerID","SummonerLevel","SummonerName","SummonerKey","SummonerDescription")

return(sum\_spell\_df)

}

sum\_spell\_df <- sum\_spell\_prep(sum\_spell\_info\_list)

Note: No R-code Development required for Business Questions 1.

Business Question #2 – R-code

The below code answers the following question, Who is the champion with the highest win rate? Lowest win rate? (The below Section uses library ggplot2)

#Two important functions: Create a function that takes in a champion ID and returns the number of games they were present in.

gamesCount = function(x){

totalGamesVector= c(games\_df$t1\_champ1id[games\_df$t1\_champ1id==x], games\_df$t1\_champ2id[games\_df$t1\_champ2id==x], games\_df$t1\_champ3id[games\_df$t1\_champ3id==x], games\_df$t1\_champ4id[games\_df$t1\_champ4id==x], games\_df$t1\_champ5id[games\_df$t1\_champ5id==x], games\_df$t2\_champ1id[games\_df$t2\_champ1id==x], games\_df$t2\_champ2id[games\_df$t2\_champ2id==x], games\_df$t2\_champ3id[games\_df$t2\_champ3id==x], games\_df$t2\_champ4id[games\_df$t2\_champ4id==x], games\_df$t2\_champ5id[games\_df$t2\_champ5id==x])

totalGameCount = length(totalGamesVector)

return(totalGameCount)

}

#Create a function that takes in a champion ID and returns the win rate for that champion.

winRate = function(x) {

totalGames = gamesCount(x)

totalWinsVector = c(games\_df$t1\_champ1id[games\_df$winner==1 & games\_df$t1\_champ1id == x], games\_df$t1\_champ2id[games\_df$winner==1 & games\_df$t1\_champ2id == x], games\_df$t1\_champ3id[games\_df$winner==1 & games\_df$t1\_champ3id == x], games\_df$t1\_champ4id[games\_df$winner==1 & games\_df$t1\_champ4id == x], games\_df$t1\_champ5id[games\_df$winner==1 & games\_df$t1\_champ5id == x], games\_df$t2\_champ1id[games\_df$winner==2 & games\_df$t2\_champ1id == x], games\_df$t2\_champ2id[games\_df$winner==2 & games\_df$t2\_champ2id == x], games\_df$t2\_champ3id[games\_df$winner==2 & games\_df$t2\_champ3id == x], games\_df$t2\_champ4id[games\_df$winner==2 & games\_df$t2\_champ4id == x], games\_df$t2\_champ5id[games\_df$winner==2 & games\_df$t2\_champ5id == x])

totalWins = length(totalWinsVector)

totalWinRate = (totalWins / totalGames)\*100

return(totalWinRate)

}

# Win rate per champion

max(unlist(champ\_df$ChampID))

min(unlist(champ\_df$ChampID))

#Add winrate to champ dataframe

for (x in 1:516){

champ\_df$winrate[champ\_df$ChampID==x] <- winRate(x)

}

#Highest Win Rate

champ\_df$ChampionName[which.max(champ\_df$winrate)]

#Lowest Win rate

champ\_df$ChampionName[which.min(champ\_df$winrate)]

#Win rate plot

library("ggplot2")

#Add Champion Names as rownames

row.names(champ\_df) <- champ\_df$ChampionName

row.names(champ\_df)

#Create a dataframe with the top 5 champions

top5 <- head(champ\_df[order(-(champ\_df$winrate)), ], 5)

ggplot(top5, aes(x=row.names(top5), y=winrate)) + geom\_col(color="black", fill="springgreen") + ggtitle("Bar Chart: Highest Win Rate Champions") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(size = 3) ) + geom\_text(aes(label=top5$winrate), vjust=-0.3, size=3.5)

#Create a dataframe with the bottom 5 champions

bot5 <- head(champ\_df[order((champ\_df$winrate)), ], 5)

ggplot(bot5, aes(x=row.names(bot5), y=winrate)) + geom\_col(color="black", fill="springgreen") + ggtitle("Bar Chart: Lowest Win Rate Champions") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(size = 3) ) + geom\_text(aes(label=bot5$winrate), vjust=-0.3, size=3.5)

#Overall Plot

ggplot(champ\_df, aes(x=row.names(champ\_df), y=winrate)) + geom\_col(color="black", fill="pink") + coord\_flip() + ggtitle("Bar Chart: Win Rate per Champion") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(size = 3) )

#Split the plot between above and below average win rates

aboveaveragechamps <- champ\_df[champ\_df$winrate > mean(champ\_df$winrate),]

belowaveragechamps <- champ\_df[champ\_df$winrate < mean(champ\_df$winrate),]

ggplot(aboveaveragechamps, aes(x=row.names(aboveaveragechamps), y=winrate)) + geom\_col(color="black", fill="pink") + coord\_flip() + ggtitle("Bar Chart: Win Rate per Above Average Champion") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(hjust=1, size = 6.5) )

ggplot(belowaveragechamps, aes(x=row.names(belowaveragechamps), y=winrate)) + geom\_col(color="black", fill="pink") + coord\_flip() + ggtitle("Bar Chart: Win Rate per Below Average Champion") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(hjust=1, size = 6.5) )

#top 25 and bottom 25 champs

quantile(champ\_df$winrate, c(0.25, 0.75))

top25champs <- champ\_df[champ\_df$winrate > quantile(champ\_df$winrate, 0.75),]

bottom25champs <- champ\_df[champ\_df$winrate < quantile(champ\_df$winrate, 0.25),]

ggplot(top25champs, aes(x=row.names(top25champs), y=winrate)) + geom\_col(color="black", fill="pink") + coord\_flip() + ggtitle("Bar Chart: Win Rate for Top 25% Champion") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(hjust=1, size =8))

ggplot(bottom25champs, aes(x=row.names(bottom25champs), y=winrate)) + geom\_col(color="black", fill="pink") + coord\_flip() + ggtitle("Bar Chart: Win Rate Bottom 25% Champion") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Win Rate (%)") + xlab("Champion Name") + theme(axis.text.y = element\_text(hjust=1, size = 8) )

Business Question #3 – R-code

The below code answers the following question, What does the highest win rate champion lose to the most? (The below Section uses library ggplot2)

#\*\*Take the instances where the champion with the highest

#win rate lost to a set of champions and tally those losses

#or put those champions into a vector\*\*

#Champ ID with highest win rate

highwinratechampid <- as.numeric(unlist(champ\_df$ChampID[which.max(champ\_df$winrate)]))

#Generate a list Team 2 champions who won a game against a team with at least one high win rate champion

T1bestchampcounter <-

as.vector(rbind(games\_df$t2\_champ1id[games\_df$winner==2 & (games\_df$t1\_champ1id==highwinratechampid | games\_df$t1\_champ2id==highwinratechampid| games\_df$t1\_champ3id==highwinratechampid| games\_df$t1\_champ4id==highwinratechampid| games\_df$t1\_champ5id==highwinratechampid)],

games\_df$t2\_champ2id[games\_df$winner==2 & (games\_df$t1\_champ1id==highwinratechampid | games\_df$t1\_champ2id==highwinratechampid| games\_df$t1\_champ3id==highwinratechampid| games\_df$t1\_champ4id==highwinratechampid| games\_df$t1\_champ5id==highwinratechampid)],

games\_df$t2\_champ3id[games\_df$winner==2 & (games\_df$t1\_champ1id==highwinratechampid | games\_df$t1\_champ2id==highwinratechampid| games\_df$t1\_champ3id==highwinratechampid| games\_df$t1\_champ4id==highwinratechampid| games\_df$t1\_champ5id==highwinratechampid)],

games\_df$t2\_champ4id[games\_df$winner==2 & (games\_df$t1\_champ1id==highwinratechampid | games\_df$t1\_champ2id==highwinratechampid| games\_df$t1\_champ3id==highwinratechampid| games\_df$t1\_champ4id==highwinratechampid| games\_df$t1\_champ5id==highwinratechampid)],

games\_df$t2\_champ5id[games\_df$winner==2 & (games\_df$t1\_champ1id==highwinratechampid | games\_df$t1\_champ2id==highwinratechampid| games\_df$t1\_champ3id==highwinratechampid| games\_df$t1\_champ4id==highwinratechampid| games\_df$t1\_champ5id==highwinratechampid)]))

#Team 2's Best Counter to the Highest Win Rate Champion

champ\_df$ChampionName[champ\_df$ChampID==mfv(T1bestchampcounter)]

#Generate a list Team 1 champions who won a game against a team with at least one high win rate champion

T2bestchampcounter <-

as.vector(rbind(games\_df$t1\_champ1id[games\_df$winner==1 & (games\_df$t2\_champ1id==highwinratechampid | games\_df$t2\_champ2id==highwinratechampid| games\_df$t2\_champ3id==highwinratechampid| games\_df$t2\_champ4id==highwinratechampid| games\_df$t2\_champ5id==highwinratechampid)],

games\_df$t1\_champ2id[games\_df$winner==1 & (games\_df$t2\_champ1id==highwinratechampid | games\_df$t2\_champ2id==highwinratechampid| games\_df$t2\_champ3id==highwinratechampid| games\_df$t2\_champ4id==highwinratechampid| games\_df$t2\_champ5id==highwinratechampid)],

games\_df$t1\_champ3id[games\_df$winner==1 & (games\_df$t2\_champ1id==highwinratechampid | games\_df$t2\_champ2id==highwinratechampid| games\_df$t2\_champ3id==highwinratechampid| games\_df$t2\_champ4id==highwinratechampid| games\_df$t2\_champ5id==highwinratechampid)],

games\_df$t1\_champ4id[games\_df$winner==1 & (games\_df$t2\_champ1id==highwinratechampid | games\_df$t2\_champ2id==highwinratechampid| games\_df$t2\_champ3id==highwinratechampid| games\_df$t2\_champ4id==highwinratechampid| games\_df$t2\_champ5id==highwinratechampid)],

games\_df$t1\_champ5id[games\_df$winner==1 & (games\_df$t2\_champ1id==highwinratechampid | games\_df$t2\_champ2id==highwinratechampid| games\_df$t2\_champ3id==highwinratechampid| games\_df$t2\_champ4id==highwinratechampid| games\_df$t2\_champ5id==highwinratechampid)]))

#Team 1's Best Counter to Team 1's Highest Win Rate Champion

champ\_df$ChampionName[champ\_df$ChampID==mfv(T2bestchampcounter)]

#The best counter overall

champ\_df$ChampionName[champ\_df$ChampID==mfv(c(T1bestchampcounter, T2bestchampcounter))]

#Generate the histogram to view the counter ranking against Janna

counter\_rank1\_df <- count(T1bestchampcounter)

counter\_rank1\_df$team <- rep(1, dim(counter\_rank1\_df)[1])

counter\_rank2\_df <- count(T2bestchampcounter)

counter\_rank2\_df$team <- rep(2, dim(counter\_rank2\_df)[1])

counter\_rank\_df <- rbind(counter\_rank1\_df, counter\_rank2\_df)

colnames(counter\_rank\_df) <- c('id', 'freq', 'team')

counter\_rank\_df$name <- unlist(champ\_df$ChampionName[match(counter\_rank\_df$id, champ\_df$ChampID)])

row.names(counter\_rank\_df) <- NULL

top\_5\_ids <- count(c(T1bestchampcounter, T2bestchampcounter))

top\_5\_ids <- top\_5\_ids[order(-top\_5\_ids$freq),][1:5,]

top5\_1\_df <- counter\_rank\_df[which(counter\_rank\_df$team==1),][match(top\_5\_ids$x, counter\_rank\_df$id),]

top5\_2\_df <- counter\_rank\_df[which(counter\_rank\_df$team==2),][match(top\_5\_ids$x, counter\_rank\_df$id),]

top\_5\_df <- rbind(top5\_1\_df, top5\_2\_df)

top\_5\_df$team[top\_5\_df$team==1] <- 'team\_1'

top\_5\_df$team[top\_5\_df$team==2] <- 'team\_2'

ggplot(data=top\_5\_df, aes(x=reorder(name, -freq), y=freq, fill=team)) + geom\_bar(stat="identity", width=0.8, position=position\_dodge()) + geom\_text(aes(label=freq), hjust=2, color="white", position = position\_dodge(0.5), size=4) + ggtitle("Top Five Best Counter") + xlab("Champ\_Name") + ylab("Count\_Time") + coord\_flip()

Business Question #4 – R-code

The below code answers the following question, What are the top two summoner spells for the champion with the most wins?

#Take the champion with the highest win rate and record all the summoner spells used for that champion in a vector.

bestChampSpell1 = c( games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1id == bestChamp], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2id == bestChamp], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3id == bestChamp], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4id == bestChamp], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5id == bestChamp], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1id == bestChamp], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2id == bestChamp], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3id == bestChamp], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4id == bestChamp], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5id == bestChamp]

)

#Do the same for spell number 2.

bestChampSpell2 = c( games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1id == bestChamp], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2id == bestChamp], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3id == bestChamp], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4id == bestChamp], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5id == bestChamp], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1id == bestChamp], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2id == bestChamp], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3id == bestChamp], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4id == bestChamp], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5id == bestChamp]

)

#Mode of summoner spell 1 vector and summoner spell 2 vector will return the most frequently taken summoner spells.

mfv(bestChampSpell1)

#ID is 4 for Flash.

mfv(bestChampSpell2)

#ID is 3 for Exhaust.

#This is okay, but summoner spell number is really just the position of the spell on one of two keys.

#sum1 could be ID 4 and sum2 could be ID 3, or vice versa, and they would be the exact same spells.

#We need to combine the summoner spells into one big vector and find the requency.

bestChampSpells = c(bestChampSpell1, bestChampSpell2)

#Now we can find the frequency. In fact, let's create a function that we can use later to find the frequency of a vector of summoner spells.

#This function can only be used on the entire listing of summoner spells within both columns. This will not produce accurate results for a single column of summoner spells.

#This is fine, however, as a single columns is useless to our calculation.

sumSpellFreq = function(v){

SpellID = unique(v)

counts = tabulate(match(v, SpellID))

percentUsed = (counts/length(v))\*2\*100 #multiply this value by 2 here since there are 2 summoner spells per game.

df1 = data.frame(SpellID, counts, percentUsed)

df = merge(df1, summonerlist, by = "SpellID")

return(data.frame(SpellName = df$SpellName, SpellCount = df$counts, PercentTaken = paste(round(df$percentUsed, 2),"%")))

}

JannaSpells = sumSpellFreq(bestChampSpells)

JannaSpells

#Now we can see that Flash is taken 99.85% of the time and exhaust is taken 93% of the time.

#Using the vector creation from above, we can create a function that will receive a champ ID and return a vector that contains all the summoner spells used by a champion.

#We can use this function to help with other questions.

sumSpellList = function(x){

c(games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1id == x], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2id == x], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3id == x], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4id == x], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5id == x], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1id == x], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2id == x], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3id == x], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4id == x], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5id == x], games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1id == x], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2id == x], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3id == x], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4id == x], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5id == x], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1id == x], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2id == x], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3id == x], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4id == x], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5id == x])

}

#Graph of Janna's spell percentage.

JannaSpellsPlot = grid.table(JannaSpells)

JannaSpellsPlot

Business Question #5 – R-code

The below code answers the following question, What is the best composition of champions? (The below Section uses library ggplot2)

# use ifelse function to collect the ids from the team that won the game

win\_champs <- data.frame(ifelse(games\_df$winner == 1, games\_df$t1\_champ1id, games\_df$t2\_champ1id), ifelse(games\_df$winner == 1, games\_df$t1\_champ2id, games\_df$t2\_champ2id), ifelse(games\_df$winner == 1, games\_df$t1\_champ3id, games\_df$t2\_champ3id), ifelse(games\_df$winner == 1, games\_df$t1\_champ4id, games\_df$t2\_champ4id), ifelse(games\_df$winner == 1, games\_df$t1\_champ5id, games\_df$t2\_champ5id))

colnames(win\_champs) <- c('champ1id', 'champ2id', 'champ3id', 'champ4id', 'champ5id')

# takes around 2 mins to sum up the team ids as a list

for (i in 1:dim(win\_champs)[1]){

champs\_v <- c(win\_champs$champ1id[i], win\_champs$champ2id[i], win\_champs$champ3id[i], win\_champs$champ4id[i], win\_champs$champ5id[i])

champs\_v <- sort(champs\_v)

win\_champs$team\_ids[i] <- list(champs\_v)

print(i/dim(win\_champs)[1])

}

# a beet way to count the occurrence is to conver the list to string

for (i in 1:dim(win\_champs)[1]){

win\_champs$str\_team\_ids[i] <- paste(win\_champs$team\_ids[i], sep=" ")

print(i/dim(win\_champs)[1])

}

# count the frequency for each team config in string data type

freq <- count(win\_champs, vars = 'str\_team\_ids')

# match the frequency and string team ids

win\_champs$team\_freq <- freq$freq[match(win\_champs$str\_team\_ids, freq$str\_team\_ids)]

# get teams which won the most times

top\_teams <- win\_champs$team\_ids[win\_champs$team\_freq == max(win\_champs$team\_freq)

# set up a function to map each id to its name

id2name <- function(team\_ids){

name\_lists <- champ\_df$ChampionName[match(unlist(team\_ids), champ\_df$ChampID)]

return(c(names(unlist(name\_lists))))

}

# clean the result and save it to a data frame

top\_team\_df <- data.frame()

for (i in 1:length(top\_teams)){

ids = top\_teams[i]

names = list(id2name(top\_teams[i]))

top\_team\_df[i,1] <- list(ids)

top\_team\_df[i,2] <- list(names)

}

colnames(top\_team\_df) <- c('ids', 'names')

View(top\_team\_df)

# create a champs ranking graph among the top teams

champ\_rank\_df <- count(unlist(top\_team\_df$names))

colnames(champ\_rank\_df) <- c('name', 'freq')

top\_10\_cooperator <- champ\_rank\_df[order(-champ\_rank\_df$freq),][1:10,]

ggplot(data=top\_10\_cooperator, aes(x=reorder(name, freq), y=freq, fill=name)) + geom\_bar(stat="identity", width=0.8, position=position\_dodge()) + geom\_text(aes(label=freq), hjust=2, color="white",position = position\_dodge(0.5), size=4) + ggtitle("Top 10 Cooperator") + xlab("Champ\_Name") + ylab("Win\_Frequency") + coord\_flip()

|  |
| --- |
| Name list of each team |
| c("Jax", "Twitch", "LeeSin", "Lux", "Syndra") |
| c("Singed", "MonkeyKing", "Kayn", "Braum", "Jhin") |
| c("Soraka", "Akali", "Lux", "Khazix", "Jhin") |
| c("MasterYi", "JarvanIV", "Yorick", "Jinx", "Thresh") |
| c("Urgot", "Tristana", "LeeSin", "Braum", "Zed") |
| c("MissFortune", "Akali", "Leona", "Sejuani", "AurelionSol") |
| c("Sivir", "Swain", "Leona", "Lux", "Khazix") |
| c("Sivir", "Swain", "Leona", "Lux", "Khazix") |
| c("Gragas", "Fiora", "Kayn", "Jhin", "Thresh") |
| c("LeeSin", "Riven", "KogMaw", "Zed", "Nami") |
| c("Urgot", "Tristana", "Sona", "LeeSin", "Cassiopeia") |
| c("MasterYi", "Jax", "Talon", "Rakan", "Xayah") |
| c("Jax", "Twitch", "LeeSin", "Lux", "Syndra") |
| c("Tryndamere", "Twitch", "Orianna", "Sejuani", "Thresh") |
| c("Caitlyn", "Maokai", "Akali", "Zed", "Ornn") |
| c("Tristana", "Blitzcrank", "Renekton", "KogMaw", "Kindred") |
| c("Gragas", "Fiora", "Kayn", "Jhin", "Thresh") |
| c("Tristana", "Blitzcrank", "Renekton", "KogMaw", "Kindred") |
| c("Urgot", "Tristana", "LeeSin", "Braum", "Zed") |
| c("Soraka", "Akali", "Lux", "Khazix", "Jhin") |
| c("Teemo", "Shaco", "Janna", "Brand", "Varus") |
| c("Orianna", "MonkeyKing", "Brand", "Jhin", "RekSai") |
| c("Teemo", "Shaco", "Janna", "Brand", "Varus") |
| c("Rammus", "Riven", "Ahri", "Rakan", "Xayah") |
| c("LeeSin", "Riven", "KogMaw", "Zed", "Nami") |
| c("Soraka", "Akali", "Lux", "Khazix", "Jhin") |
| c("Orianna", "MonkeyKing", "Brand", "Jhin", "RekSai") |
| c("Anivia", "Caitlyn", "Leona", "Fiora", "Ornn") |
| c("Urgot", "Tristana", "Sona", "LeeSin", "Cassiopeia") |
| c("Morgana", "Malphite", "Hecarim", "Yasuo", "Jinx") |
| c("Morgana", "Malphite", "Hecarim", "Yasuo", "Jinx") |
| c("MasterYi", "Jax", "Talon", "Rakan", "Xayah") |
| c("Caitlyn", "Maokai", "Akali", "Zed", "Ornn") |
| c("LeeSin", "Riven", "KogMaw", "Zed", "Nami") |
| c("DrMundo", "Orianna", "Kayn", "Rakan", "Xayah") |
| c("Morgana", "Malphite", "Hecarim", "Yasuo", "Jinx") |
| c("Gangplank", "Sejuani", "Velkoz", "Rakan", "Xayah") |
| c("Gragas", "Fiora", "Kayn", "Jhin", "Thresh") |
| c("Tryndamere", "Twitch", "Orianna", "Sejuani", "Thresh") |
| c("Teemo", "Shaco", "Janna", "Brand", "Varus") |
| c("DrMundo", "Orianna", "Kayn", "Rakan", "Xayah") |
| c("DrMundo", "Orianna", "Kayn", "Rakan", "Xayah") |
| c("Urgot", "Tristana", "Sona", "LeeSin", "Cassiopeia") |
| c("MissFortune", "Taric", "Lissandra", "Kayn", "RekSai") |
| c("MissFortune", "Akali", "Leona", "Sejuani", "AurelionSol") |
| c("MissFortune", "Akali", "Leona", "Sejuani", "AurelionSol") |
| c("Anivia", "Caitlyn", "Leona", "Fiora", "Ornn") |
| c("Rammus", "Riven", "Ahri", "Rakan", "Xayah") |
| c("Anivia", "Caitlyn", "Leona", "Fiora", "Ornn") |
| c("MasterYi", "JarvanIV", "Yorick", "Jinx", "Thresh") |
| c("MasterYi", "JarvanIV", "Yorick", "Jinx", "Thresh") |
| c("MasterYi", "Jax", "Talon", "Rakan", "Xayah") |
| c("Caitlyn", "Maokai", "Akali", "Zed", "Ornn") |
| c("Rammus", "Riven", "Ahri", "Rakan", "Xayah") |
| c("Gangplank", "Sejuani", "Velkoz", "Rakan", "Xayah") |
| c("Singed", "MonkeyKing", "Kayn", "Braum", "Jhin") |
| c("Urgot", "Tristana", "LeeSin", "Braum", "Zed") |
| c("Tristana", "Blitzcrank", "Renekton", "KogMaw", "Kindred") |
| c("MissFortune", "Taric", "Lissandra", "Kayn", "RekSai") |
| c("Tryndamere", "Twitch", "Orianna", "Sejuani", "Thresh") |
| c("Gangplank", "Sejuani", "Velkoz", "Rakan", "Xayah") |
| c("MissFortune", "Taric", "Lissandra", "Kayn", "RekSai") |
| c("Sivir", "Swain", "Leona", "Lux", "Khazix") |
| c("Singed", "MonkeyKing", "Kayn", "Braum", "Jhin") |
| c("Jax", "Twitch", "LeeSin", "Lux", "Syndra") |
| c("Orianna", "MonkeyKing", "Brand", "Jhin", "RekSai") |

Business Question #6 – R-code

The below code answers the following question, What is the most banned champion? What is their win rate? (The below Section uses library ggplot2)

#Create a vector of all the banned champion.

#Note that both sides can have the same bans, so a champion can be banned by both sides in a match.

allBans <- c(games\_df$t1\_ban1,games\_df$t1\_ban2,games\_df$t1\_ban3,games\_df$t1\_ban4,games\_df$t1\_ban5,games\_df$t2\_ban1,games\_df$t2\_ban2,games\_df$t2\_ban3,games\_df$t2\_ban4,games\_df$t2\_ban5)

#Find the mode of the banned champions. This will be the most banned champion.

mfv(allBans)

#The most banned champion is ID 157, which is Yasuo.

#Win rate of most banned champion.

winRate(157) #50.15% win rate.

#Calculate ban rate. Create a function that takes in a champion ID and returns a ban rate.

banRate = function (ID){

length(allBans[allBans == ID])/nrow(games\_df)\*100

}

#Observe the ban rate of the most banned champion.

banRate(157)

#Create a data frame with ban rates for all champs.

#Use lapply with the banRate function to return the ban rates for the champions.

#At the same time, we can unlist this result and create a data frame that matches the champion ID.

champBanRateList = data.frame(champList, unlist(lapply(champList, banRate)))

#Let's clean up the column names. We'll be using this data frame a bunch.

colnames(champBanRateList) = c("ChampID", "BanRate")

#Merge champion win rate list with champ list.

champBanRateList = merge(champBanRateList, champlist, by = "ChampID")

champBanRateList = data.frame(ChampID = champBanRateList$ChampID, ChampName = champBanRateList$ChampionName, BanRate = champBanRateList$BanRate)

#Find the highest win rate. Use that index to find the champion ID.

champBanRateList[which.max(champBanRateList[,3]),]

#The champion with the highest ban rate is ID 157, Yasuo, with a 64.12% ban rate.

champBanRateList[which.min(champBanRateList[,3]),]

#The champion with the lowest ban rate is ID 72, Skarner, with a 0.14% ban rate.

#DataFrame with champs, win rate, and ban rate.

ChampInfo = merge(champBanRateList, champWinRateList, by = "ChampID")

ChampInfo = data.frame(ChampID = ChampInfo$ChampID, ChampName = ChampInfo$ChampName.x, WinRate = round(ChampInfo$WinRate, 2), BanRate = round(ChampInfo$BanRate, 2))

#Plot the ban rate.

BanBar = ggplot(ChampInfo, aes(x = ChampName, y = BanRate)) + geom\_col(color = "black", fill = "blue") + coord\_flip() + ggtitle("Bar Chart: Ban Rate for Champions") + theme(plot.title = element\_text(hjust = 0.5)) + theme(axis.text.y = element\_text(hjust=0, size =3))

BanBar

#Ban Rate for Top 10 Champions with win rate.

Champinfo2 = ChampInfo[order(-ChampInfo$BanRate),]

Champinfo2 = Champinfo2[1:10,]

Champinfo2 = melt(Champinfo2, id.vars = "ChampName", measure.vars = c("WinRate", "BanRate"))

BanBar2 = ggplot(Champinfo2, aes(x = ChampName, y = value, fill = variable, label = paste0(value,"%"))) + geom\_bar(stat = "identity") + coord\_flip() + ggtitle("Bar Chart: Top 10 Banned Champs with Win Rate") + theme(plot.title = element\_text(hjust = 0.5)) + theme(axis.text.y = element\_text(face = "bold", hjust=0, size =9)) + geom\_text(size = 4, position = position\_stack(vjust = 0.5), color = "white") + scale\_fill\_manual(values=c("dark green", "dark red"))

BanBar2

Business Question #7 – R-code

The below code answers the following question, Which objective kill count is the best predictor of winner? (The below Section uses library ggplot2)

#Scale the objectives and create a new data frame with the information.

compareObjectives <- data.frame(games\_df$winner,scale(games\_df$t1\_towerKills, center = FALSE, scale = TRUE), scale(games\_df$t1\_inhibitorKills, center = FALSE, scale = TRUE),scale(games\_df$t1\_baronKills, center = FALSE, scale = TRUE), scale(games\_df$t1\_dragonKills, center = FALSE, scale = TRUE),scale(games\_df$t1\_riftHeraldKills, center = FALSE, scale = TRUE),scale(games\_df$t2\_towerKills, center = FALSE, scale = TRUE), scale(games\_df$t2\_inhibitorKills, center = FALSE, scale = TRUE),scale(games\_df$t2\_baronKills, center = FALSE, scale = TRUE),scale(games\_df$t2\_dragonKills, center = FALSE, scale = TRUE),scale(games\_df$t2\_riftHeraldKills, center = FALSE, scale = TRUE)

)

#Rename the columns with easily accessible names

colnames(compareObjectives) <- c("winner", "t1\_towerKills", "t1\_inhibitorKills", "t1\_baronKills", "t1\_dragonKills", "t1\_riftHeraldKills","t2\_towerKills", "t2\_inhibitorKills", "t2\_baronKills", "t2\_dragonKills", "t2\_riftHeraldKills")

#Plot to get an example of the data.

plot(compareObjectives$t1\_towerKills, compareObjectives$winner)

#Looks like linear regression will not work here. Use logit or probit.

#Set winner outcome to binary to perform logit analysis. Blue becomes 0, Red becomes 1.

compareObjectives$winner = ifelse (compareObjectives$winner == 2, 1, 0)

#Call the logit and probits models into variables.

ObjLog = glm(formula = winner ~ ., family = binomial(logit), data = compareObjectives)

#Look at sumaries of the data.

summary(ObjLog)

#For both models, tower kills were the most important factors and dragon kills did not matter. However, we need to perform a VIF to see if any factors have multicollinearity.

vif(ObjLog)

#Drop dragons.

ObjLog2 = glm(formula = winner ~ t1\_towerKills + t1\_inhibitorKills + t1\_baronKills + t1\_riftHeraldKills + t2\_towerKills + t2\_inhibitorKills +t2\_baronKills + t2\_riftHeraldKills , family = binomial(logit), data = compareObjectives)

summary(ObjLog2)

vif(ObjLog2)

#Use the logit model to predict the winner. The type is response to ensure values between 1 and 0. Round the output to get values between 1 and 0.

compareObjectives$predWinner = round(predict(ObjLog2, type = "resp"))

#Percentage Correct

perc\_log <- length(which(compareObjectives$winner==compareObjectives$predWinner))/dim(compareObjectives)[1]

perc\_log

#Confusion matrix using the logit model.

#Create a table for the actual and predicted value counts.

bq6results <- table(Actual = compareObjectives$winner, Pred = compareObjectives$predWinner)

bq6plot = qplot(1:2, 1:2) + theme\_void() + annotation\_custom(grob = tableGrob(bq6results)) + annotate("text", x = 1.515, y = 1.65, label = "Prediction", fontface = 2) + annotate("text", x = 1.38, y = 1.57, label = "Actual", fontface = 2) + annotate("text", x = 1.515, y = 1.35, label = "Percentage Correct - 95.97%", colour = "dark green", fontface = 2)

bq6plot

#Graph the prediction based on the analysis.

ObjLogPlot = ggplot(compareObjectives, aes(x = t2\_towerKills, y = winner)) + stat\_smooth(method="glm", method.args=list(family="binomial"), se=TRUE)

ObjLogPlot

#Add a histogram to the plot.

logi.hist.plot(compareObjectives$t2\_towerKills,compareObjectives$winner,boxp=FALSE,type="hist",col="gray")

Business Question # 8 – R-code

The below code answers the following question, Which team has a higher win rate?

# draw pie chart for team win rate

team\_win\_count <- c(length(games\_df$winner[games\_df$winner==1]), length(games\_df$winner[games\_df$winner==2]))

team\_win\_rate <- team\_win\_count / length(row.names(games\_df))

team\_label <- c('Team\_One', 'Team\_Two')

percent <- function(x, digits = 2, format = "f", ...) {

paste0(formatC(100 \* x, format = format, digits = digits, ...), "%")

}

precent\_team\_win\_rate <- c(percent(team\_win\_rate[1]), percent(team\_win\_rate[2]))

team\_win\_rate\_str <- c(toString(precent\_team\_win\_rate[1]), toString(precent\_team\_win\_rate[2]))

chart\_label <- c()

for (i in 1:2){

chart\_label[i] <- paste(c(team\_label[i], team\_win\_rate\_str[i]), collapse=" - ")

}

pie3D(team\_win\_rate, labels=chart\_label, col=c('blue', 'red'), explode=0.1,main="Pie Chart of Team Win Rate")

Business Question #9 – R-code

The below code answers the following question, Which first to indicator is the best predictor of winner?

#Which champion is present in games with the highest percentage of that indicator?

#Make a dataframe for winners and first to indicators.

compareFirst <- data.frame(games\_df$winner,games\_df$firstBlood, games\_df$firstTower, games\_df$firstInhibitor, games\_df$firstBaron, games\_df$firstDragon, games\_df$firstRiftHerald

)

#Rename the columns.

colnames(compareFirst) = c("winner", "firstBlood", "firstTower", "firstInhibitor", "firstBaron", "firstDragon", "firstRiftHerald")

#We will be using logit/probit again, so we need to set the proper result values.

compareFirst$winner = ifelse (compareFirst$winner == 2, 1, 0)

#Run logit analysis on these features.

firstLM = lm(formula = winner ~ ., data = compareFirst)

summary(firstLM)

#Check for multicollinearity.

vif(firstLM)

#First inhibitor is the predictor with the largest coefficient value. Rift herald appears to not be significant at all.

#Remove rift herald and rerun the regression.

secondLM = lm(formula = winner ~ firstBlood + firstTower + firstInhibitor + firstBaron + firstDragon, data = compareFirst)

summary(secondLM)

#Try to predict the winner given the first to.

compareFirst$predWinner = round(predict(secondLM, type = "resp"))

#Percentage Correct

perc\_log2 <- length(which(compareFirst$winner==compareFirst$predWinner))/dim(compareFirst)[1]

perc\_log2

#Confusion matrix using the logit model.

#Create a table for the actual and predicted value counts.

bq8results <- table(Actual = compareFirst$winner, Pred = compareFirst$predWinner)

bq8plot = qplot(1:2, 1:2) + theme\_void() + annotation\_custom(grob = tableGrob(bq8results)) + annotate("text", x = 1.515, y = 1.65, label = "Prediction", fontface = 2) + annotate("text", x = 1.38, y = 1.57, label = "Actual", fontface = 2) + annotate("text", x = 1.515, y = 1.35, label = "Percentage Correct - 82.96%", colour = "dark green", fontface = 2)

bq8plot

Business Question #10 – R-code

The below code answers the following question, Which champion (or team composition) is in games with the most objectives? (The below Section uses library ggplot2)

#Compile objectives for team 1

games\_df$T1CompiledObjectives[games\_df$winner==1] <- ( scale(games\_df$t1\_towerKills[games\_df$winner==1], center = FALSE, scale = TRUE) + scale(games\_df$t1\_inhibitorKills[games\_df$winner==1], center = FALSE, scale = TRUE) +scale(games\_df$t1\_baronKills[games\_df$winner==1], center = FALSE, scale = TRUE) +scale(games\_df$t1\_dragonKills[games\_df$winner==1], center = FALSE, scale = TRUE) +scale(games\_df$t1\_riftHeraldKills[games\_df$winner==1], center = FALSE, scale = TRUE) )

#Return the champ with max compiled objectives for team 1

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t1\_champ1id[which.max(games\_df$T1CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t1\_champ2id[which.max(games\_df$T1CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t1\_champ3id[which.max(games\_df$T1CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t1\_champ4id[which.max(games\_df$T1CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t1\_champ5id[which.max(games\_df$T1CompiledObjectives)]]

#Compile objectives for team 2

games\_df$T2CompiledObjectives[games\_df$winner==2] <- (scale(games\_df$t2\_towerKills[games\_df$winner==2], center = FALSE, scale = TRUE) + scale(games\_df$t2\_inhibitorKills[games\_df$winner==2], center = FALSE, scale = TRUE) + scale(games\_df$t2\_baronKills[games\_df$winner==2], center = FALSE, scale = TRUE) + scale(games\_df$t2\_dragonKills[games\_df$winner==2], center = FALSE, scale = TRUE) + scale(games\_df$t2\_riftHeraldKills[games\_df$winner==2], center = FALSE, scale = TRUE) )

#Return the champ with max compiled objectives for team 2

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t2\_champ1id[which.max(games\_df$T2CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t2\_champ2id[which.max(games\_df$T2CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t2\_champ3id[which.max(games\_df$T2CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t2\_champ4id[which.max(games\_df$T2CompiledObjectives)]]

champ\_df$ChampionName[champ\_df$ChampID==games\_df$t2\_champ5id[which.max(games\_df$T2CompiledObjectives)]]

#Generate a new table with the top teams with most objectives for Team 1

t1co <- head(games\_df[order(-(games\_df$T1CompiledObjectives)), ], 10)

#Add the name of champion to the table

#t1 c1

t1con <- t1co$t1\_champ1id

for (x in t1con) {

t1co$t1\_champ1name[t1co$t1\_champ1id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#c1 c2

t1con <- t1co$t1\_champ2id

for (x in t1con) {

t1co$t1\_champ2name[t1co$t1\_champ2id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#t1 c3

t1con <- t1co$t1\_champ3id

for (x in t1con) {

t1co$t1\_champ3name[t1co$t1\_champ3id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#t1 c4

t1con <- t1co$t1\_champ4id

for (x in t1con) {

t1co$t1\_champ4name[t1co$t1\_champ4id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#t1 c5

t1con <- t1co$t1\_champ5id

for (x in t1con) {

t1co$t1\_champ5name[t1co$t1\_champ5id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

rownames(t1co) <- paste(t1co$t1\_champ1name, t1co$t1\_champ2name, t1co$t1\_champ3name, t1co$t1\_champ4name,t1co$t1\_champ5name, sep="\n")

#Generate a plot

ggplot(t1co, aes(x=row.names(t1co), y=T1CompiledObjectives))+ geom\_col(color="black", fill="skyblue") + ggtitle("Bar Chart: Top 10 Team 1 Compositions with Most Objective Kills") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Objective Total") + xlab("Team Composition") + theme(axis.text.y = element\_text(hjust=1, size = 6.5) )

#Generate a similar plot for Team 2

#Generate a new table with the top teams with most objectives for Team 2

t2co <- head(games\_df[order(-(games\_df$T2CompiledObjectives)), ], 10)

#Add the name of champion to the table

#t2 c1

t2con <- t2co$t2\_champ1id

for (x in t2con) {

t2co$t2\_champ1name[t2co$t2\_champ1id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#c1 c2

t2con <- t2co$t2\_champ2id

for (x in t2con) {

t2co$t2\_champ2name[t2co$t2\_champ2id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x]

}

#t2 c3

t2con <- t2co$t2\_champ3id

for (x in t2con) {

t2co$t2\_champ3name[t2co$t2\_champ3id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

#t2 c4

t2con <- t2co$t2\_champ4id

for (x in t2con) {

t2co$t2\_champ4name[t2co$t2\_champ4id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x]

}

#t2 c5

t2con <- t2co$t2\_champ5id

for (x in t2con) {

t2co$t2\_champ5name[t2co$t2\_champ5id==x] <- unlist(champ\_df$ChampionName[champ\_df$ChampID==x])

}

rownames(t2co) <- paste(t2co$t2\_champ1name, t2co$t2\_champ2name, t2co$t2\_champ3name, t2co$t2\_champ4name,t2co$t2\_champ5name, sep="\n")

#Generate a plot

ggplot(t2co, aes(x=row.names(t2co), y=T2CompiledObjectives))+ geom\_col(color="black", fill="darkred") + ggtitle("Bar Chart: Top 10 Team 2 Compositions with Most Objective Kills") + theme(plot.title = element\_text(hjust = 0.5)) + ylab("Objective Total") + xlab("Team Composition") + theme(axis.text.y = element\_text(hjust=1, size = 6.5

Business Question #11– R-code

The below code answers the following question, Summoner Spell Win Rates

#Possible spells list. Exclude all spells that are not in summoner's rift and Clarity.

spellList = summonerlist$SpellID[c(1:7,9,10)]

#Number of times where a spell was chosen by a player.

spellTotal = function(x){

totalGames = c(games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1\_sum1==x], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2\_sum1==x], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3\_sum1==x], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4\_sum1==x], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5\_sum1==x], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1\_sum1==x], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2\_sum1==x], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3\_sum1==x], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4\_sum1==x], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5\_sum1==x], games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1\_sum2==x], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2\_sum2==x], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3\_sum2==x], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4\_sum2==x], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5\_sum2==x], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1\_sum2==x], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2\_sum2==x], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3\_sum2==x], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4\_sum2==x], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5\_sum2==x]

)

return(length(totalGames))

}

#Percentage of time summoner taken by users. 10 Possoble players, 51490 possible instances of players.

spellPercent = function (x){

totalGames = c(games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1\_sum1==x], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2\_sum1==x], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3\_sum1==x], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4\_sum1==x], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5\_sum1==x], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1\_sum1==x], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2\_sum1==x], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3\_sum1==x], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4\_sum1==x], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5\_sum1==x], games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1\_sum2==x], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2\_sum2==x], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3\_sum2==x], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4\_sum2==x], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5\_sum2==x], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1\_sum2==x], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2\_sum2==x], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3\_sum2==x], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4\_sum2==x], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5\_sum2==x]

)

percentage = length(totalGames)/(51490\*10)\*100

return(percentage)

}

#All percentages of posible summoner spell choices.

SpellFreqTable = data.frame(SpellID = spellList,SpellFreq = unlist(lapply(spellList, spellPercent)))

SpellFreqTable = merge(SpellFreqTable, summonerlist, by = "SpellID")

SpellFreqTable = SpellFreqTable[,c(1,4,2)]

SpellFreqTable$SpellFreq = paste(round(SpellFreqTable$SpellFreq,2), "%")

SpellFreqTable

#Plot the summoner spell frequency table.

grid.table(SpellFreqTable)

#Summoner Win Rate by Summoner Spell.

SpellChampWin = function(c, s) {

#Number of times a summoner spell, s, appeared in a game with champion, c.

totaltimestaken = c(games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1id == c & games\_df$t1\_champ1\_sum1 == s], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2id == c & games\_df$t1\_champ2\_sum1 == s], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3id == c & games\_df$t1\_champ3\_sum1 == s], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4id == c & games\_df$t1\_champ4\_sum1 == s], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5id == c & games\_df$t1\_champ5\_sum1 == s], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1id == c & games\_df$t2\_champ1\_sum1 == s], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2id == c & games\_df$t2\_champ2\_sum1 == s], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3id == c & games\_df$t2\_champ3\_sum1 == s], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4id == c & games\_df$t2\_champ4\_sum1 == s], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5id == c & games\_df$t2\_champ5\_sum1 == s], games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1id == c & games\_df$t1\_champ1\_sum1 == s], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2id == c & games\_df$t1\_champ2\_sum1 == s], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3id == c & games\_df$t1\_champ3\_sum1 == s], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4id == c & games\_df$t1\_champ4\_sum1 == s], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5id == c & games\_df$t1\_champ5\_sum1 == s], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1id == c & games\_df$t1\_champ1\_sum1 == s], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2id == c & games\_df$t2\_champ2\_sum1 == s], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3id == c & games\_df$t3\_champ3\_sum1 == s], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4id == c & games\_df$t4\_champ4\_sum1 == s], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5id == c & games\_df$t5\_champ5\_sum1 == s] )

#Number of times a summoner spell, s, appeared in game with champion, c, and the champion won.

totaltimeswon = c(games\_df$t1\_champ1\_sum1[games\_df$t1\_champ1id == c & games\_df$t1\_champ1\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ2\_sum1[games\_df$t1\_champ2id == c & games\_df$t1\_champ2\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ3\_sum1[games\_df$t1\_champ3id == c & games\_df$t1\_champ3\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ4\_sum1[games\_df$t1\_champ4id == c & games\_df$t1\_champ4\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ5\_sum1[games\_df$t1\_champ5id == c & games\_df$t1\_champ5\_sum1 == s & games\_df$winner == 1], games\_df$t2\_champ1\_sum1[games\_df$t2\_champ1id == c & games\_df$t2\_champ1\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ2\_sum1[games\_df$t2\_champ2id == c & games\_df$t2\_champ2\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ3\_sum1[games\_df$t2\_champ3id == c & games\_df$t2\_champ3\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ4\_sum1[games\_df$t2\_champ4id == c & games\_df$t2\_champ4\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ5\_sum1[games\_df$t2\_champ5id == c & games\_df$t2\_champ5\_sum1 == s & games\_df$winner == 2], games\_df$t1\_champ1\_sum2[games\_df$t1\_champ1id == c & games\_df$t1\_champ1\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ2\_sum2[games\_df$t1\_champ2id == c & games\_df$t1\_champ2\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ3\_sum2[games\_df$t1\_champ3id == c & games\_df$t1\_champ3\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ4\_sum2[games\_df$t1\_champ4id == c & games\_df$t1\_champ4\_sum1 == s & games\_df$winner == 1], games\_df$t1\_champ5\_sum2[games\_df$t1\_champ5id == c & games\_df$t1\_champ5\_sum1 == s & games\_df$winner == 1], games\_df$t2\_champ1\_sum2[games\_df$t2\_champ1id == c & games\_df$t1\_champ1\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ2\_sum2[games\_df$t2\_champ2id == c & games\_df$t2\_champ2\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ3\_sum2[games\_df$t2\_champ3id == c & games\_df$t3\_champ3\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ4\_sum2[games\_df$t2\_champ4id == c & games\_df$t4\_champ4\_sum1 == s & games\_df$winner == 2], games\_df$t2\_champ5\_sum2[games\_df$t2\_champ5id == c & games\_df$t5\_champ5\_sum1 == s & games\_df$winner == 2] )

#Find the percentage won by dividing the length of the times the champion won by the length of appearances.

percentwon = length(totaltimeswon)/length(totaltimestaken)\*100

return(round(percentwon,2))

}

#Create a function that makes a table of Summoner spell and win rate when given a Champion ID.

ChampSpellFreq = function(x) {

#Prepare the champion name for pasting later.

champpick = champlist$ChampionName[champlist$ChampID == x]

#Create a dummy vector to store the SpellChampWin results.

pblock = NULL

#Find the win percentage of all the summoner spells in question through iteration.

for (s in 1:length(SpellFreqTable$SpellID))

{

pblock = c(pblock, SpellChampWin(x, SpellFreqTable$SpellID[s]))

}

#Create a data frame with the spell names and the frequency taken.

df = data.frame(name = SpellFreqTable$SpellName, freq = paste(pblock, "%") )

#Rename the columns.

colnames(df) = c(paste(champpick,"SpellName"), paste(champpick,"SpellFreq"))

return(df)

}

YasuoSpellWinRate = ChampSpellFreq(157)

YasuoSpellWinRate

JannaSpellWinRate = ChampSpellFreq(40)

JannaSpellWinRate

#Plot the tables.

grid.table(YasuoSpells)

grid.table(JannaSpellWinRate)

Business Question #12 – R-code

The below code answers the following question, Which side gets more objectives in matches? ((The below Section uses library kernlab, e1071, ggplot2, gridExtra)

if (mean(games\_df$T1CompiledObjectives, na.rm=TRUE) > mean(games\_df$T2CompiledObjectives, na.rm=TRUE)) print("Team 1 gets more objectives") else print("Team 2 gets more objectives")

#Generate a pie chart to compare the objectives

a <- mean(games\_df$T1CompiledObjectives, na.rm=TRUE)

b <- mean(games\_df$T2CompiledObjectives, na.rm=TRUE)

slices <- c(a, b)

lbls <- c("Team 1 Compiled Objectives", "Team 2 Compiled Objectives")

pct <- round(slices/sum(slices)\*100, digits=2)

lbls <- paste(lbls, pct)

lbls <- paste(lbls,"%",sep="")

pie(slices, labels = lbls, main="Average Compiled Objectives for Team 1 and Team 2", col = c("blue", "red"))

#Compile objectives for team 1

games\_df$allobjt1 <- (scale(games\_df$t1\_towerKills, center = FALSE, scale = TRUE) + scale(games\_df$t1\_inhibitorKills, center = FALSE, scale = TRUE) +scale(games\_df$t1\_baronKills, center = FALSE, scale = TRUE) + scale(games\_df$t1\_dragonKills, center = FALSE, scale = TRUE) + scale(games\_df$t1\_riftHeraldKills, center = FALSE, scale = TRUE) )

#Compile objectives for team 2

games\_df$allobjt2 <- ( scale(games\_df$t2\_towerKills, center = FALSE, scale = TRUE) + scale(games\_df$t2\_inhibitorKills, center = FALSE, scale = TRUE) + scale(games\_df$t2\_baronKills, center = FALSE, scale = TRUE) + scale(games\_df$t2\_dragonKills, center = FALSE, scale = TRUE) + scale(games\_df$t2\_riftHeraldKills, center = FALSE, scale = TRUE)

#Create a separate dataframe with the kill objective and scale them as appropriate

bokt <- data.frame(games\_df$winner,games\_df$allobjt1,games\_df$allobjt2)

#Rename the columns for objective kill table

colnames(bokt) <- c("winner", "Team1CompiledObjectives", "Team2CompiledObjectives")

#Convert

# Team 1 == 1

# Team 2 == 0

bokt$winner <- ifelse(bokt$winner==2, 0, 1)

#covert winner to factor

bokt$winner <- as.factor(bokt$winner)

#Create a random index by sampling all indices in our dataset

random <- sample(1:dim(bokt)[1])

#Take a small subset of the data to analyze. R stalls/crashes when using more than 5000 training data

random <- random[1:7500]

bokt <- bokt[random, ]

#reset the rownames/index

rownames(bokt) <- NULL

#Re-create the random index based on our smaller dataset

random <- sample(1:dim(bokt)[1])

#Create a cut point to separate our dataset into 2/3 Train and 1/3 Test

cutpoint <- floor(2\*(dim(bokt)[1])/3)

cutpoint

#Create the train data

traindata <- bokt[random[1:cutpoint], ]

#Create the test data

testdata <- bokt[random[(cutpoint+1):dim(bokt)[1]], ]

#Build a ksvm model

ksvmmodel <- ksvm(winner~., data=traindata, kernal="rbfdot", kpar="automatic", C=10, cross=10, prob.model=TRUE)

str(ksvmmodel)

#test the model

ksvmpred <- predict(ksvmmodel, testdata)

#generate a comparison table

comptable <- data.frame(testdata$winner, ksvmpred)

#rename the columns

colnames(comptable) <- c("test", "pred")

#percentage of winners that was correctly predicted

percent <- (length(which(comptable$test==comptable$pred)))/(dim(comptable)[1])\*100

percent

#plot the results of ksvm function

#determine if prediction is "correct" or "wrong" for each case

comptable$correct <- ifelse(comptable$test==comptable$pred,"correct","wrong")

#create a new dataframe with the data for the plot

plotdata <- data.frame(comptable$correct, testdata$Team1CompiledObjectives, testdata$Team2CompiledObjectives, testdata$winner, comptable$pred)

#rename the columns

colnames(plotdata) <- c("Correct", "Team1CompiledObjectives", "Team2CompiledObjectives", "Winner", "Predict")

#create a plot for the ksvm results

ksvmplot <- ggplot(data=plotdata, aes(x=Team1CompiledObjectives, y=Team2CompiledObjectives)) + geom\_point(data=plotdata, aes(size=Correct, color=Winner, shape=Predict)) + ggtitle("Winner by Team 1&2 Compiled Objectives ksvm Plot")

ksvmplot

#ksvm Confusion Matrix

ksvmresults <- table(test=comptable$test, pred=comptable$pred)

print(ksvmresults)

#build a model using naive bayes

nb <- naiveBayes(winner~., data=traindata)

#test the model

nbpred <- predict(nb, testdata)

#generate a comparison table

nbcomptable <- data.frame(testdata$winner, nbpred)

#rename the columns

colnames(nbcomptable) <- c("test", "pred")

#percentage of winners that was correctly predicted

nbpercent <- (length(which(nbcomptable$test==nbcomptable$pred)))/(dim(nbcomptable)[1])\*100

nbpercent

#plot the results of the nb function, determine if prediction is "correct" or "wrong" for each case

nbcomptable$correct <- ifelse(nbcomptable$test==nbcomptable$pred,"correct","wrong")

#create a new dataframe with the data for the plot

nbplotdata <- data.frame(nbcomptable$correct, testdata$Team1CompiledObjectives, testdata$Team2CompiledObjectives, testdata$winner, nbcomptable$pred)

#rename the columns

colnames(nbplotdata) <- c("Correct", "Team1CompiledObjectives", "Team2CompiledObjectives", "Winner", "Predict")

#create a plot for the nb results

nbplot <- ggplot(data=nbplotdata, aes(x=Team1CompiledObjectives, y=Team2CompiledObjectives)) + geom\_point(data=nbplotdata, aes(size=Correct, color=Winner, shape=Predict)) + ggtitle("Winner by Team 1&2 Compiled Objectives naiveBayes Plot")

nbplot

#naïve Bayes Confusion Matrix

nbresults <- table(test=nbcomptable$test, pred=nbcomptable$pred)

print(nbresults)

#Show both plots in one window

grid.arrange(ksvmplot, nbplot, nrow=2, top="Winnner by Team 1&2 Objective Differential by Comparison")

Business Question #13 – R-code

The below code answers the following question, How much is Rift Herald an indicator of win chance?

#There can only be one Rift Herald taken per game

# If Team 1 wins and they get the rift herald

q <-length(games\_df$gameId[games\_df$winner==1 & games\_df$t1\_riftHeraldKills==1])

# If Team 2 wins and they get the rift herald

w <- length(games\_df$gameId[games\_df$winner==2 & games\_df$t2\_riftHeraldKills==1])

# If Team 1 wins and they get no rift herald

e <- length(games\_df$gameId[games\_df$winner==1 & games\_df$t1\_riftHeraldKills==0])

# If Team 2 wins and they get no rift herald

r <- length(games\_df$gameId[games\_df$winner==2 & games\_df$t2\_riftHeraldKills==0])

#Pie Chart Comparing the different Win Rates

aslices <- c(q,w,e,r)

albls <- c("Team 1 Win Rate with Rift Herald", "Team 2 Win Rate with Rift Herald", "Team 1 Win Rate without Rift Herald", "Team 2 Win Rate without Rift Herald")

apct <- round(aslices/sum(aslices)\*100, digits=2)

albls <- paste(albls, apct, sep="\n")

albls <- paste(albls,"%",sep="")

pie(aslices, labels = albls, main="Team 1 and Team 2 Win Percentages given Rift Herald Kill", col = c("skyblue", "darkred", "blue", "red"))

#Another Pie Chart Comparing the different Win Rates

aaslices <- c(q+w, e+r)

aalbls <- c("Win Percentage with Rift Herald", "Win Percentage without Rift Herald")

aapct <- round(aaslices/sum(aaslices)\*100, digits=2)

aalbls <- paste(aalbls, aapct, sep="\n")

aalbls <- paste(aalbls,"%",sep="")

pie(aaslices, labels = aalbls, main="Win Percentages given Rift Herald Kill", col = c("blue", "red"))

Business Question #14 – R-code

The below code answers the following question, Create a function that takes a champion name as an input and returns a win rate for that champion and the top champion they lose to. (this answer uses the library("modeest"))

nemesis <- function(cn){

#Convert name into champion ID

y <- (champ\_df$ChampID[champ\_df$ChampionName==cn])[1]

n <- c(games\_df$t1\_champ1id[games\_df$winner==2 & games\_df$t1\_champ1id!=y], games\_df$t1\_champ2id[games\_df$winner==2 & games\_df$t1\_champ2id!=y], games\_df$t1\_champ3id[games\_df$winner==2 & games\_df$t1\_champ3id!=y], games\_df$t1\_champ4id[games\_df$winner==2 & games\_df$t1\_champ4id!=y], games\_df$t1\_champ5id[games\_df$winner==2 & games\_df$t1\_champ5id!=y], games\_df$t2\_champ1id[games\_df$winner==1 & games\_df$t2\_champ1id!=y], games\_df$t2\_champ2id[games\_df$winner==1 & games\_df$t2\_champ2id!=y], games\_df$t2\_champ3id[games\_df$winner==1 & games\_df$t2\_champ3id!=y], games\_df$t2\_champ4id[games\_df$winner==1 & games\_df$t2\_champ4id!=y], games\_df$t2\_champ5id[games\_df$winner==1 & games\_df$t2\_champ5id!=y])

n <- mfv(n)

n <- champ\_df$ChampionName[champ\_df$ChampID==n]

return(paste(n, "is the nemesis of", cn))

}

#Test

nemesis("Tristana")

nemesis("Annie")

nemesis("Ahri")

nemesis("Jinx")

Business Question #15 – R-code

The below code answers the following question, Create a function that takes a champion name as an input and returns the summoner spell combination with the highest win rate. (The below Section uses library modeest)

champspellwincombo <- function(cn){

#Convert name into champion ID

y <- (champ\_df$ChampID[champ\_df$ChampionName==cn])[1]

sum1wins<- c(games\_df$t1\_champ1\_sum1[games\_df$winner==1 & games\_df$t1\_champ1id==y], games\_df$t1\_champ2\_sum1[games\_df$winner==1 & games\_df$t1\_champ2id==y], games\_df$t1\_champ3\_sum1[games\_df$winner==1 & games\_df$t1\_champ3id==y], games\_df$t1\_champ4\_sum1[games\_df$winner==1 & games\_df$t1\_champ4id==y], games\_df$t1\_champ5\_sum1[games\_df$winner==1 & games\_df$t1\_champ5id==y], games\_df$t2\_champ1\_sum1[games\_df$winner==2 & games\_df$t2\_champ1id==y], games\_df$t2\_champ2\_sum1[games\_df$winner==2 & games\_df$t2\_champ2id==y], games\_df$t2\_champ3\_sum1[games\_df$winner==2 & games\_df$t2\_champ3id==y], games\_df$t2\_champ4\_sum1[games\_df$winner==2 & games\_df$t2\_champ4id==y], games\_df$t2\_champ5\_sum1[games\_df$winner==2 & games\_df$t2\_champ5id==y]) #Summoner slot 1

sum1wins <- mfv(sum1wins)

sum1wins <- sum\_spell\_df$SummonerName[sum\_spell\_df$SummonerID==sum1wins]

sum2wins<- c(games\_df$t1\_champ1\_sum2[games\_df$winner==1 & games\_df$t1\_champ1id==y], games\_df$t1\_champ2\_sum2[games\_df$winner==1 & games\_df$t1\_champ2id==y], games\_df$t1\_champ3\_sum2[games\_df$winner==1 & games\_df$t1\_champ3id==y], games\_df$t1\_champ4\_sum2[games\_df$winner==1 & games\_df$t1\_champ4id==y], games\_df$t1\_champ5\_sum2[games\_df$winner==1 & games\_df$t1\_champ5id==y], games\_df$t2\_champ1\_sum2[games\_df$winner==2 & games\_df$t2\_champ1id==y], games\_df$t2\_champ2\_sum2[games\_df$winner==2 & games\_df$t2\_champ2id==y], games\_df$t2\_champ3\_sum2[games\_df$winner==2 & games\_df$t2\_champ3id==y], games\_df$t2\_champ4\_sum2[games\_df$winner==2 & games\_df$t2\_champ4id==y], games\_df$t2\_champ5\_sum2[games\_df$winner==2 & games\_df$t2\_champ5id==y]) #Summoner Slot 2

sum2wins <- mfv(sum2wins)

sum2wins <- sum\_spell\_df$SummonerName[sum\_spell\_df$SummonerID==sum2wins]

return(paste(cn, "wins the most with", sum1wins, "and", sum2wins))

}

champspellwincombo("Tristana")#Test

champspellwincombo("Ahri") #Test

champspellwincombo("Jinx") #Test

champspellwincombo("Caitlyn") #Test

Business Question #16 – R-code

The below code answers the following question, Create a function that takes a champion name as an input and returns the champion they win the most with. (The below Section uses library modeest)

champpartner <- function(cn){

#Convert name into champion ID

y <- (champ\_df$ChampID[champ\_df$ChampionName==cn])[1]

duo <- c(games\_df$t1\_champ1id[games\_df$winner==1 & games\_df$t1\_champ1id!=y], games\_df$t1\_champ2id[games\_df$winner==1 & games\_df$t1\_champ2id!=y], games\_df$t1\_champ3id[games\_df$winner==1 & games\_df$t1\_champ3id!=y], games\_df$t1\_champ4id[games\_df$winner==1 & games\_df$t1\_champ4id!=y], games\_df$t1\_champ5id[games\_df$winner==1 & games\_df$t1\_champ5id!=y], games\_df$t2\_champ1id[games\_df$winner==2 & games\_df$t2\_champ1id!=y], games\_df$t2\_champ2id[games\_df$winner==2 & games\_df$t2\_champ2id!=y], games\_df$t2\_champ3id[games\_df$winner==2 & games\_df$t2\_champ3id!=y], games\_df$t2\_champ4id[games\_df$winner==2 & games\_df$t2\_champ4id!=y], games\_df$t2\_champ5id[games\_df$winner==2 & games\_df$t2\_champ5id!=y])

duo <- mfv(duo)

duo <- champ\_df$ChampionName[champ\_df$ChampID==duo]

return(paste(cn, "wins the most with", duo))

}

#Test

champpartner("Tristana")

champpartner("Ahri")

champpartner("Jinx")

champpartner("Caitlyn")