Character Class Victory Conditions in League of Legends

Ethan Swartzentruber

12/10/2019

Abstract

In the computer game League of Legends, players choose from 146 different characters at the start of the game. These characters are divided into six different categories by the game's developer, Riot Games (often simply called Riot). This analysis uses logistic regression in an attempt to determine to what extent this division is valid - in particular, whether the division of characters presented by Riot captures a concrete difference in victory conditions of the character classes. Although the conventional wisdom amongst the League of Legends community is that these character classes should represent a stark difference in victory condition, this analysis failed to find a statistical basis for the existence of such a difference.

Introduction/Background

League of Legends is an extremely popular 5v5 competitive strategy game. Players join a queue to be matched with other players of a comparable skill level, and at the beginning of each game, choose from a roster of 146 (as of Dec 10 2019) playable characters. Each of these characters is unique, and in the ranked 5v5 format (the format which this analysis uses data from), only one player can choose a given character in each game. These characters are divided into six classes by Riot Games, the developer and publisher of League of Legends: Assassin, Fighter, Mage, Marksman, Support, and Tank. These classes differ greatly in aesthetic theme and in how they feel to the player. For example, some classes are built for one versus one duels with enemy players, and some are built to thrive in conflicts involving the entirity of both teams. The conventional wisdom amongst the League of Legends community is that these character classes differ in the degree to which they are *individualistic* - the degree to which it is required that the character be played well, rather than simply be on a team which has some collective advantage, in order to win.

There are several different types of concrete advantage to be had in League of Legends that this study considers. The first is *gold*. Gold is granted in some quantity by almost every activity in League of Legends, and is used to buy items, which make the players' characters more powerful. This is one of two individual progression systems in League of Legends. The second type of individual advantage is *experience*. Experience points are granted to players who are standing in a radius of neutral or enemy units when they die, and occasionally through several other mechanisms which are specific to certain characters. Hitting thresholds of experience causes your character to level up, which makes your character more powerful.

In addition to individual advantage, there are two types of collective advantage which this analysis considers. The first is *neutral monsters*. These are large monsters which are in limited supply in a game of League of Legends - once a team has killed the monster, the other team cannot kill it for some amount of time before it respawns. Each of these monsters grants a permanent power bonus to every player on the team which killed it. The second type of collective advantage is *destroyed buildings*. Both teams start with a set of structures on their side of the map, which provide defense. These buildings can be destroyed by the opposing team, and most of them do not respawn. Players win the game either by destroying a building at the center of the enemy team's base, or by the enemy team surrendering (both are common).

Data

Data Collection

The 8064 games which comprise the data for this analysis was collected via Riot Games' public API. Usage of this API involved first requesting the list of games for a specific player, and then processing each of those games. In order to maintain stability in the underlying patterns in gameplay, I chose to sample names from a continuous section of the League of Legends leaderboard - all the games in the dataset are played by players of comparable rank. Because Riot Games does not publish an official leaderboard that is easily scrapable, I used a third party leaderboard from the website https://www.op.gg/ . This leaderboard is extremely close to the official leaderboard - it is determined by the players' in-game ranking and is mostly up to date. I chose to sample from just inside the 95th percentile of players, because ranks much higher than that caused the population of players to shrink to the point where the observations were no longer independent, because the same games were appearing in the histories of multiple players in the sample.

Data Dictionary

- **assists** (Numeric): How many times the player assisted in a kill during the game
- **ccScore** (Numeric): Number of seconds the player spent inhibiting the movement of enemy players in the game, weighted by number of players and type of movement inhibiting effect (this figure is provided by Riot API)
- **championClass** (Categorical): Which class of character the player was playing during the game
- **deaths** (Numeric): How many times the player died during the game

- **firstBaron** (Categorical): Which team killed the first Baron, a late game neutral monster
- **firstBloodKill** (Binary): Whether the player got the first kill of the game
- **firstDragon** (Categorical): Which team killed the first Dragon, a neutral monster
- **firstTower** (Categorical): Which team destroyed the first tower, a defensive building
- **fiveMinuteGoldDelta** (Numeric): Difference between the player of interest's gold total at 5 minutes and the average for the game
- **fiveMinuteXPDelta** (Numeric): Difference between the player of interest's experience total at 5 minutes and the average for the game
- **goldEarned** (Numeric): Total amount of gold earned by the player during the game
- **kills** (Numeric): How many kills the player achieved during the game
- **visionScore** (Numeric): An aggregated measure of how well the player tracks enemy movement around the map, provided by Riot Games API
- **wardKills** (Numeric): Number of "wards" (units which observe the enemy players to gather information) killed by the player
- **win** (Binary Response): Whether the player of interest won the game

Model

Exploratory Data Analysis

Boxplotting against some of the types of advantage discussed in the background section, it appears we should expect a relatively small separation in predicted winrate based on goldEarned and the five minute deltas, and a relatively large separation based on dragon kills (plot included in appendix).

Binned plots looking for interactions between championClass and other predictors come up looking like there isn't much of an interaction. In particular, such a plot with championClass and five minute gold delta shows relationships that are similar for every class of character - this runs *highly* contrary to the conventional wisdom in the League of Legends community - many players believe that five minute gold delta should matter quite a bit for Assassin and Fighter, and almost not at all for Support and Tank (plot included in appendix).

Another interesting lack of interaction is in a metric called *vision score* - this is known as one of the primary

responsibilities of support players, and as such would be expected to show a stronger relationship with support than it would with other character classes. However, the relationship again appears similar (plot included in appendix).

Model Specification

Several of the predictors in the dataset showed very high correlation with the response, and thus were too correlated with the other predictors - they caused problems with multicollinearity. They are as follows:

- goldEarned - Typically in a game of League of Legends, only the winning team is able to destroy buldings in the inner parts of the enemy team's base - games are simply not often that close. Because the object of the game is to destroy a building at the center of the enemy team's base, and because destroying buildings grants gold to all members of the team that destroyed it, this final influx of gold makes goldEarned track the outcome of the game very well. In place of goldEarned, I decided to include only the five minute gold delta - this gives information about the usefulness of gold as an advantage to each character class, but doesn't fall prey to the fact that most other types of advantage are leveraged to obtain gold.
- Baron kill variables - "Baron" is a late game neutral monster, which gives a massive advantage to the team that kills it, and it is often used by a team to close out the game - games often end immediately after or shortly after, and teams will sometimes forfeit the moment that the enemy team kills the baron. Because of this, baron kills correlate extremely well with wins.
- Dragon kill variables - "Dragon" is a neutral monster which is available almost the entire game, respawning some time after each death. Although early game control of the dragon is very important, dragon kills later in the game are likely to be side effects of an otherwise decided game - I chose to include only firstDragon and not the dragon kill count variables.

With the remaining variables, I performed several iterations of backward selection, using different'y scoped starting models and different metrics (AIC and BIC). In the end, I chose the model produced by using AIC, with the logit function as a link. The "flat, then steep, then flat again" shape of the logit function makes intuitive sense for many of League of Legends' advantage mechanisms, as advantages typically compound on each other in a way that the playerbase calls "snowballing", and because of mechanisms in the game called "comeback mechanics" which exist to give the losing team a fighting chance. For example, in League of Legends, if you accumulate a large amount of gold more than the enemy players, killing you grants them a "shut down", which is worth a large amount of gold.

Backward selection using AIC gives a model which does not include any interaction terms. Examining the summary for the full model shows that while several of the interaction coefficients are significant at the .05 level in this model, none are significant at the .01 or .001 level, and none have noteworthy z-values. Given that the expectation from the League of Legends community would be that the differences be stark between character classes, this is not enough to make a claim for significant difference between classes. Full model summary and backwards selected model summaries are available in the appendix.

Model Assessment

The model shows a very high AUC of 0.912 (plot included in appendix) - if the purpose of this analysis was prediction, this may indicate an issue in generalizability. For the purposes of inference, however, this seems fine.

A binned plot of residuals versus predicted probability of winning shows a definite pattern, but not many points fall outside the 95% confidence interval except at very low probabilities of winning. I was tempted to conclude that these games represent times when something goes horribly wrong in the beginning of the game for the player of interest's team - having players quit is rare in League of Legends, but is known to happen, and when it does happen, it leaves the team that now has only four players with almost no path to victory. However, were this the case, I would expect to see a similar group of points at the far right of the plot - games where something similarly disastrous befell the enemy team. While the points on the far right of the

plot do follow a similar shape, there aren't nearly as many in that pattern as there are in the grouping on the left side. The reason for the innacuracy of the model at the lower extreme remains unclear to me at this time.

Conclusions

The only indication in this analysis that Riot Games' division of characters captrues a difference in victory condition is the p-value of several interaction coefficients in the full model (summary in appendix). However, these coefficients show unremarkable z-values, they fall out when using backward selection with either AIC or BIC, and their p-values themselves aren't nearly as low as those of some of the other predictors. It is safe to say that the actual statistics don't back up the League of Legends community's conventional understanding of character classes - the division put forth by Riot Games definitely captures a difference in gameplay, but does not capture a siginificant difference in victory condition amongst the advantage metrics considered in this analysis.

Other interesting insights into the game can be found from the summary of the final model. Firstly, the Fighter and Assassin character classes win more often than the other classes - these are known as the more "individualistic" classes, the characters that typically thrive in duels are small skirmishes involving one or two players from each team. Additionally, the coefficient for assists is higher magnitude than that of kills in the final model, even though assists grant a much lower reward in the game. My interpretation of this is that assists are proxying for something that League of Legends players typically call "presence" or "proximity"; roughly, this is the proportion of engagements with the enemy team happening in the game which the player is part of - being present for a high proportion of winning engagements (thereby scoring assists) requires the player be knowledgable about when and where opportunities for attack are to be had, and requires that the player manage their time well, giving themselves enough time to move across the map to be part of these engagements.

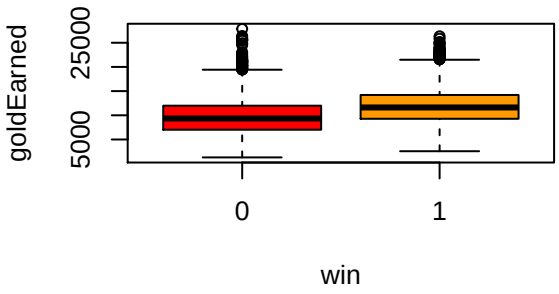
Early game collective advantages are very important - firstDragon and firstTower are very strong predictors in the model. This tracks well with the current conventional wisdom about the game - a recent change to League of Legends greatly increased the power level of the bonus granted by killing dragons, and the game has seen an uptick in the play rates of characters that thrive in the early game brawls that arise out of both teams committing a large amount of resources toward obtaining the first of these bonuses by killing the first dragon.

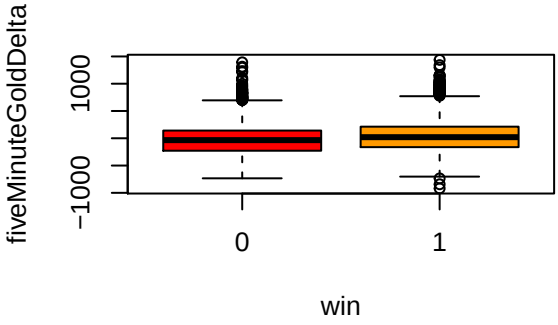
Limitations and Future Work

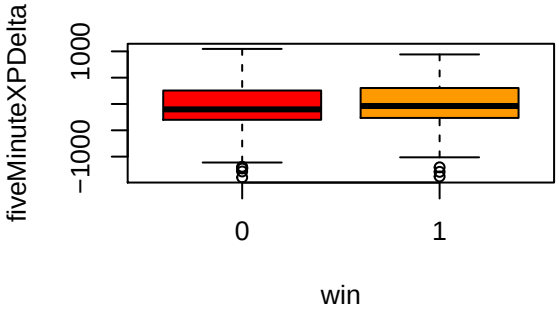
It was tempting to explore individual characters as a predictor in the model, but with the amount of data used in this analysis, there were far too many sparse categories - a hierarchical model with 146 categories might work, but even still, significantly more data would be needed - playable characters in League of Legends vary wildly in their rate of play, and so to capture more than a handful of games from a good proportion of the characters at the bottom of that distribution, a much alrger dataset would be needed. The collection of this data would be very straightforward, but would require an upgraded API key for the Riot Games API - the same data collection methods and scripts as used in this analysis could be used.

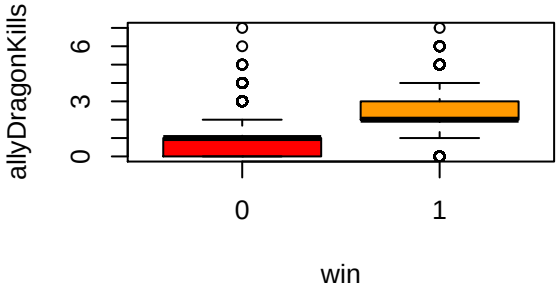
APPENDIX

EDA Boxplots

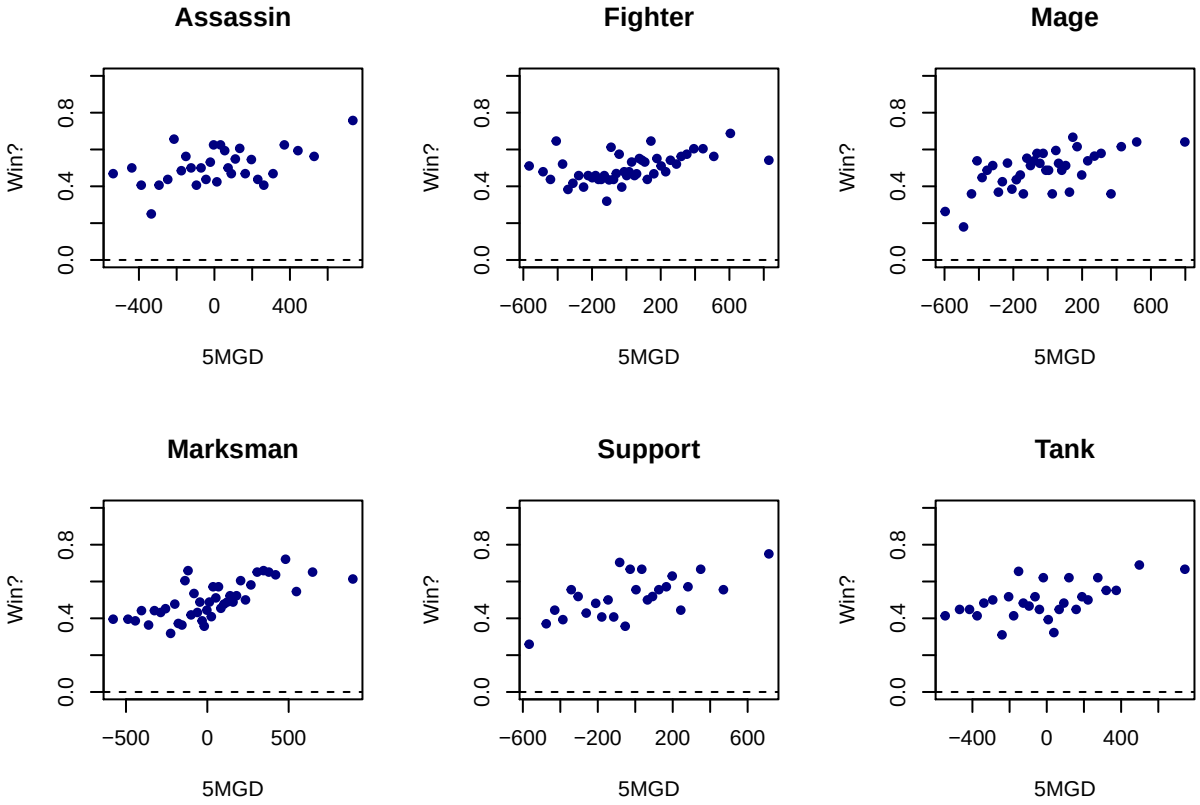




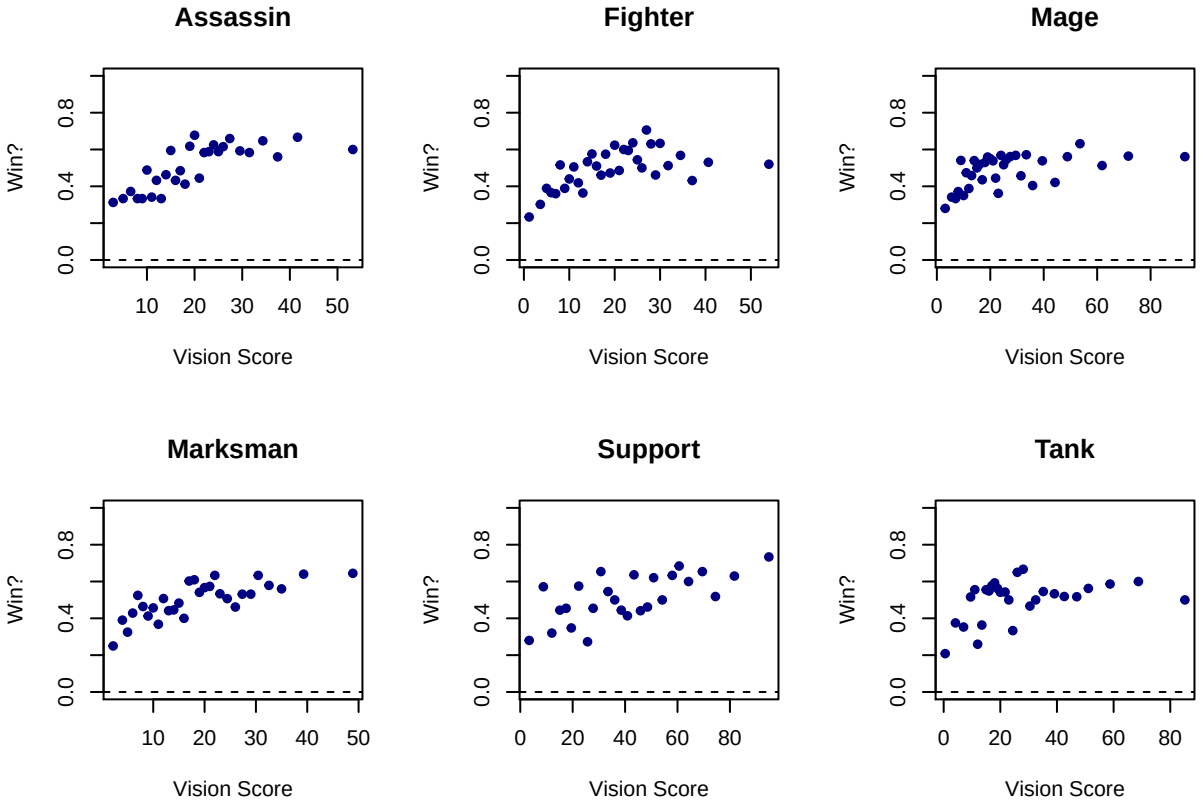




Binned Plots 5MGD by championClass



Binned Plots visionScore by championClass



Full Model Sumamry (before backward selection)

term	estimate	std.error	statistic	p.value
(Intercept)	1.2016298	0.2598349	4.6245899	0.0000038
fiveMinuteGoldDelta	-0.0000241	0.0003796	-0.0635872	0.9492989
championClassFighter	0.4399411	0.3040908	1.4467426	0.1479690
championClassMage	0.1644164	0.3244622	0.5067353	0.6123406
championClassMarksman	0.1162735	0.3125696	0.3719925	0.7098984
championClassSupport	-0.7036243	0.3784502	-1.8592258	0.0629951
championClassTank	0.4026084	0.3761426	1.0703612	0.2844568
firstBloodKillTrue	-0.2257468	0.2898703	-0.7787856	0.4361061
kills	0.1960812	0.0207241	9.4614960	0.0000000
deaths	-0.4371316	0.0359909	-12.1456009	0.0000000
assists	0.2333782	0.0083981	27.7894190	0.0000000
ccScore	-0.0007984	0.0001776	-4.4961888	0.0000069
wardKills	-0.0332734	0.0123534	-2.6934596	0.0070715
firstDragonEnemy	-1.4084473	0.1925796	-7.3135857	0.0000000
firstDragonNone	0.2961061	0.8769789	0.3376434	0.7356319
fiveMinuteXPDelta	0.0000906	0.0002873	0.3152244	0.7525913
firstTowerEnemy	-1.5492874	0.0663263	-23.3585625	0.0000000
firstTowerNone	-1.3120044	1.2895516	-1.0174113	0.3089578
championClassFighter:firstBloodKillTrue	0.1192769	0.3430209	0.3477248	0.7280468
championClassMage:firstBloodKillTrue	0.2411135	0.4033485	0.5977798	0.5499869
championClassMarksman:firstBloodKillTrue	0.1100772	0.3467983	0.3174097	0.7509327
championClassSupport:firstBloodKillTrue	-0.0489345	0.5560104	-0.0880101	0.9298687

term	estimate	std.error	statistic	p.value
championClassTank:firstBloodKillTrue	0.6921602	0.4518664	1.5317807	0.1255766
championClassFighter:kills	-0.0363289	0.0253556	-1.4327748	0.1519222
championClassMage:kills	-0.0204234	0.0280811	-0.7273016	0.4670412
championClassMarksman:kills	-0.0381419	0.0256188	-1.4888230	0.1365340
championClassSupport:kills	-0.0663964	0.0396704	-1.6737025	0.0941891
championClassTank:kills	-0.0449610	0.0378740	-1.1871199	0.2351803
championClassFighter:deaths	-0.0164856	0.0424189	-0.3886380	0.6975440
championClassMage:deaths	-0.0241006	0.0456412	-0.5280460	0.5974674
championClassMarksman:deaths	-0.0049293	0.0435168	-0.1132733	0.9098139
championClassSupport:deaths	0.0129219	0.0542446	0.2382146	0.8117147
championClassTank:deaths	-0.1025438	0.0523437	-1.9590464	0.0501074
championClassFighter:firstDragonEnemy	0.3070454	0.2286437	1.3428988	0.1793047
championClassMage:firstDragonEnemy	0.1243751	0.2447803	0.5081092	0.6113768
championClassMarksman:firstDragonEnemy	0.2259338	0.2347329	0.9625141	0.3357914
championClassSupport:firstDragonEnemy	0.7053490	0.2973231	2.3723320	0.0176762
championClassTank:firstDragonEnemy	0.2960723	0.2848977	1.0392233	0.2987009
championClassFighter:firstDragonNone	-0.3915012	1.1042028	-0.3545555	0.7229226
championClassMage:firstDragonNone	-0.1125448	1.4326683	-0.0785561	0.9373857
championClassMarksman:firstDragonNone	-1.1384309	1.0868768	-1.0474333	0.2948998
championClassSupport:firstDragonNone	0.2990950	1.2287077	0.2434224	0.8076782
championClassTank:firstDragonNone	-1.8567104	1.5609596	-1.1894673	0.2342558
fiveMinuteGoldDelta:championClassFighter	-0.0001716	0.0004448	-0.3857172	0.6997061
fiveMinuteGoldDelta:championClassMage	0.0000946	0.0004697	0.2013681	0.8404107
fiveMinuteGoldDelta:championClassMarksman	-0.0002671	0.0004440	-0.6016018	0.5474392
fiveMinuteGoldDelta:championClassSupport	0.0006584	0.0005726	1.1498108	0.2502218
fiveMinuteGoldDelta:championClassTank	-0.0002731	0.0005393	-0.5063819	0.6125886
championClassFighter:fiveMinuteXPDelta	-0.0002219	0.0003461	-0.6412332	0.5213712
championClassMage:fiveMinuteXPDelta	-0.0000035	0.0003664	-0.0096306	0.9923160
championClassMarksman:fiveMinuteXPDelta	0.0001886	0.0003523	0.5353173	0.5924304
championClassSupport:fiveMinuteXPDelta	-0.0001050	0.0004530	-0.2317304	0.8167474
championClassTank:fiveMinuteXPDelta	0.0001026	0.0004349	0.2359039	0.8135073

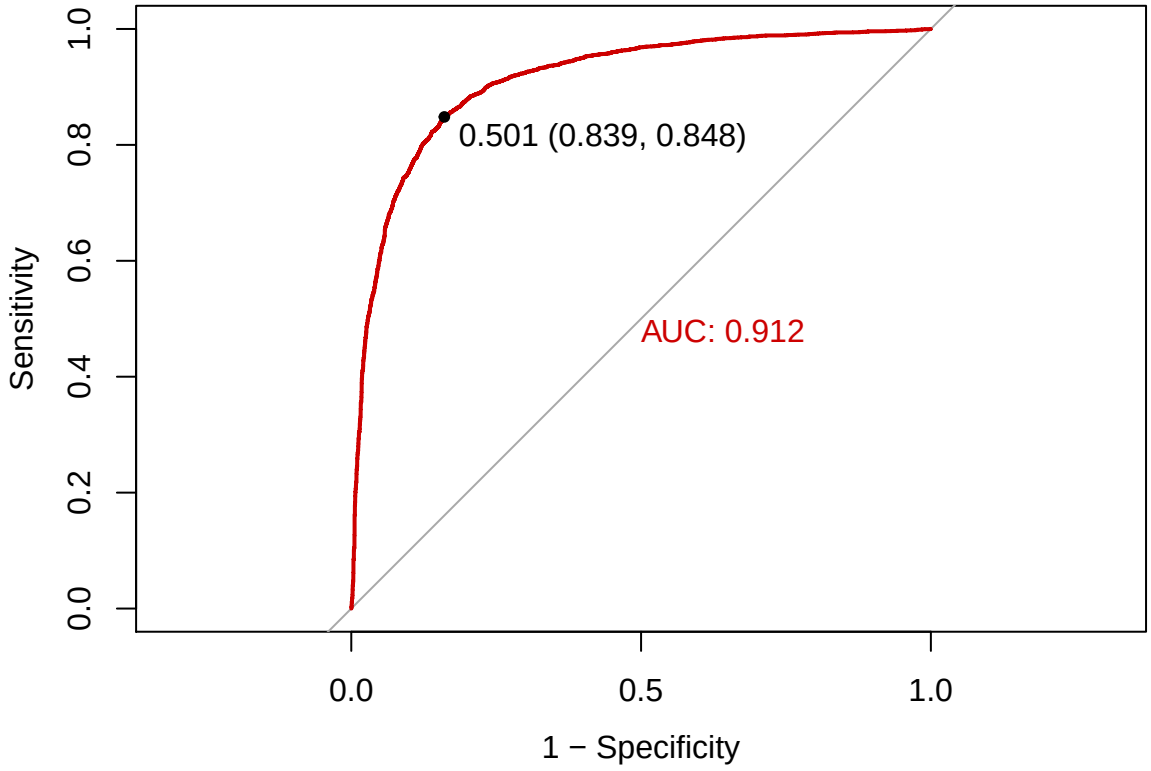
Final Model Summary (after backward selection with AIC)

term	estimate	std.error	statistic	p.value
(Intercept)	1.4316440	0.1313508	10.8993912	0.0000000
championClassFighter	0.1944502	0.1115176	1.7436724	0.0812162
championClassMage	-0.1010289	0.1209289	-0.8354411	0.4034694
championClassMarksman	-0.1416151	0.1142714	-1.2392865	0.2152394
championClassSupport	-0.6520461	0.1620143	-4.0246217	0.0000571
championClassTank	-0.2963744	0.1472335	-2.0129540	0.0441195
kills	0.1647346	0.0078182	21.0707290	0.0000000
deaths	-0.4550765	0.0135100	-33.6844731	0.0000000
assists	0.2313505	0.0082256	28.1256307	0.0000000
ccScore	-0.0007712	0.0001711	-4.5081643	0.0000065
wardKills	-0.0334499	0.0121030	-2.7637679	0.0057138
firstDragonEnemy	-1.1534423	0.0654149	-17.6327021	0.0000000
firstDragonNone	-0.2790676	0.3271181	-0.8531096	0.3935985
firstTowerEnemy	-1.5312569	0.0651395	-23.5073585	0.0000000
firstTowerNone	-1.2732831	1.2849728	-0.9909028	0.3217330

VIF Output for Final Model

Predictor	VIF
championClassFighter	2.399795
championClassMage	2.173186
championClassMarksman	2.244862
championClassSupport	1.856846
championClassTank	1.904511
kills	1.264376
deaths	1.394865
assists	1.719737
ccScore	1.175273
wardKills	1.204441
firstDragonEnemy	1.034632
firstDragonNone	1.034703
firstTowerEnemy	1.026068
firstTowerNone	1.002896

AUC Curve for final model




```
##
## Call:
## roc.default(response = games$win, predictor = fitted(model_backward_aic),     plot = T, print.thres =
##
## Data: fitted(model_backward_aic) in 4030 controls (games$win 0) < 4034 cases (games$win 1).
## Area under the curve: 0.912
```

Binnedplot vs predicted probs for final model