AIT580 – Data Analysis Individual Project

Deliverable 1 – Data Selection:

I. Dataset Metadata (size, item meanings, types)

Data was pulled from the OpenDota API for 1,007 professional Dota 2 eSports matches played between June 2 and Sep. 17 of 2020. The raw dataset in *json* format takes up 270 megabytes (MB). After my Python program parses this dataset for the information I need and saves it as a tab-separated file (tsv) its size is now only 1.1 MB.

Field.Name	**Field.Description**	**Data.Type**
match.id	unique match identifier	integer
match.id.slot	combination of match_id and player slot to get unique record key	varchar
duration.min	match length measured in minutes	decimal
radiant.win	indicates if the Radiant team won the match	logical
isRadiant	indicates if a player in a match is on the Radiant team	logical
	* team names are always Radiant (left side of game arena) and Dire (right)	
num.players	number of players in the match (sanity check field as there should always be 10)	integer
player.slot	a number encoding for a player's color, unique within a match_id	integer
kills	number of times a players kills another player's character during a match	integer
deaths	number of times a player's character dies in the match	integer
assists	number of times a player assists (does damage) to an enemy player's character that is killed	integer
gold.p.min	total amount of gold accumulated during a match divided by the match length	integer
xp.p.min	total amount of experience (leveling and character progression) during a match divided by the match length	integer
win	indicates if a player's match outcome was a win	logical
hero.dmg	total amount of damage dealt to enemy characters	integer
hero.heal	total amount of health restored to this player's character	integer
last.hits	total number of small minion (creeps) killed in a match	integer
denies	total number of enemy small minion (creeps) were killed before enemy could kill them, denying gold and xp	integer
tower.dmg	total amount of damage dealt to enemy towers (must kill specific enemy towers to win match)	integer
runes	number of times a player picked up a rune during the match	integer
	* a rune is a temporary character power-up, like running very fast or being invisible or attacking for higher damage	
obs.set	number of observer wards placed during a match	integer
	* observer wards reveal an area of the arena for a short time, giving one team a sight and information advantage	
stuns	total time (seconds) in stuns dealt by player's character to another character or minion	
first.blood	indicates if a player got the first character kill in the match	logical
match.kills	total number of kills in a match (includes both teams)	integer

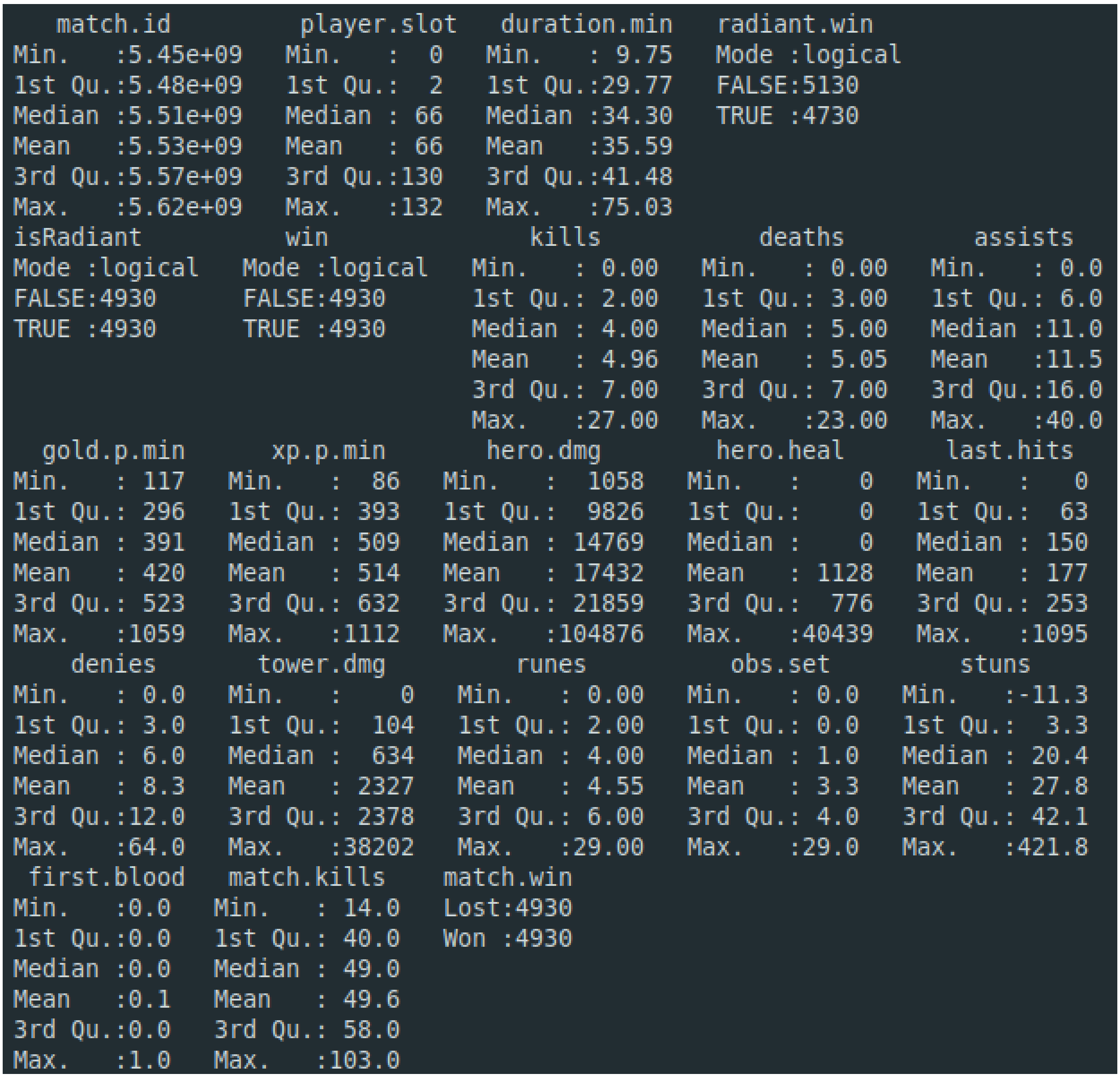
Table 1: Dataset fields - meanings and data types

Deliverable 2 – Data Analysis:

I. Descriptive Statistics and Visualizations

Summary statistics

Figure 1 below shows the summary statistics for the Dota 2 dataset including Tukey's five number summary for quantitative values – min, first quantile, median, third quantile, max – as well the median and number of NA's (if present). For a given match, the combination of *match_id* and *player_slot* (0-5 or 128-132) can be used to uniquely identify each record. We can see that most fields have a right skew to them with generally small numbers and a few larger outliers influencing the mean to be greater than the median. There are no missing values in any fields. From the *radiant_win* field, we can see that there may be a disadvantage to being on the Radiant side of the arena (left side) as more Radiant teams lost their matches.



```
    match.id          player.slot   duration.min    radiant.win    
 Min.   :5.45e+09   Min.   :  0   Min.   : 9.75   Mode :logical  
 1st Qu.:5.48e+09   1st Qu.:  2   1st Qu.:29.77   FALSE:5130     
 Median :5.51e+09   Median : 66   Median :34.30   TRUE :4730     
 Mean   :5.53e+09   Mean   : 66   Mean   :35.59                  
 3rd Qu.:5.57e+09   3rd Qu.:130   3rd Qu.:41.48                  
 Max.   :5.62e+09   Max.   :132   Max.   :75.03                  
 isRadiant          win                kills            deaths          assists    
 Mode :logical   Mode :logical   Min.   : 0.00   Min.   : 0.00   Min.   : 0.0  
 FALSE:4930      FALSE:4930      1st Qu.: 2.00   1st Qu.: 3.00   1st Qu.: 6.0  
 TRUE :4930      TRUE :4930      Median : 4.00   Median : 5.00   Median :11.0  
                                 Mean   : 4.96   Mean   : 5.05   Mean   :11.5  
                                 3rd Qu.: 7.00   3rd Qu.: 7.00   3rd Qu.:16.0  
                                 Max.   :27.00   Max.   :23.00   Max.   :40.0  
   gold.p.min        xp.p.min        hero.dmg          hero.heal         last.hits   
 Min.   : 117   Min.   :  86   Min.   :  1058   Min.   :    0   Min.   :   0  
 1st Qu.: 296   1st Qu.: 393   1st Qu.:  9826   1st Qu.:    0   1st Qu.:  63  
 Median : 391   Median : 509   Median : 14769   Median :    0   Median : 150  
 Mean   : 420   Mean   : 514   Mean   : 17432   Mean   : 1128   Mean   : 177  
 3rd Qu.: 523   3rd Qu.: 632   3rd Qu.: 21859   3rd Qu.:  776   3rd Qu.: 253  
 Max.   :1059   Max.   :1112   Max.   :104876   Max.   :40439   Max.   :1095  
     denies         tower.dmg          runes            obs.set          stuns     
 Min.   : 0.0   Min.   :    0   Min.   : 0.00   Min.   : 0.0   Min.   :-11.3  
 1st Qu.: 3.0   1st Qu.:  104   1st Qu.: 2.00   1st Qu.: 0.0   1st Qu.:  3.3  
 Median : 6.0   Median :  634   Median : 4.00   Median : 1.0   Median : 20.4  
 Mean   : 8.3   Mean   : 2327   Mean   : 4.55   Mean   : 3.3   Mean   : 27.8  
 3rd Qu.:12.0   3rd Qu.: 2378   3rd Qu.: 6.00   3rd Qu.: 4.0   3rd Qu.: 42.1  
 Max.   :64.0   Max.   :38202   Max.   :29.00   Max.   :29.0   Max.   :421.8  
  first.blood    match.kills     match.win  
 Min.   :0.0   Min.   : 14.0   Lost:4930  
 1st Qu.:0.0   1st Qu.: 40.0   Won :4930  
 Median :0.0   Median : 49.0              
 Mean   :0.1   Mean   : 49.6              
 3rd Qu.:0.0   3rd Qu.: 58.0              
 Max.   :1.0   Max.   :103.0              
```

Figure 1: Summary statistics for dataset using R summary() function

Descriptive visualizations

We can build on our dataset knowledge and do sanity checks by visually looking at frequency counts for each column. In my case, *match.kills* was originally centered around 250 with each player getting about 5 kills per match. With only 10 players in each match (50 total kills), these numbers didn't match up.. this led me to correct a bug in my program and correct the *match.kills* column. This highlights just one of many reasons to do EDA in the data analysis process.

Hero.dmg looks to have the largest range (from 0 to over 90,000), while kills has the smallest range (0- 30). As we saw in the previous summary statistics, many fields have a right skew with the majority of the data in the smaller values and with a few larger outliers to the right side. Experience per minute (*xp.p.min*) seems to be the most normally distributed, along with *match.kills* and *match.len*. From the distributions below, we can surmise that a typical match lasts about 35 minutes with 50 total kills with individual players amassing 10 assists, 5 deaths, 5 kills, 15,000 hero damage, 150 minion kills (last hits), 5 minion denies, 400 gold per minute, and 500 xp per minute.

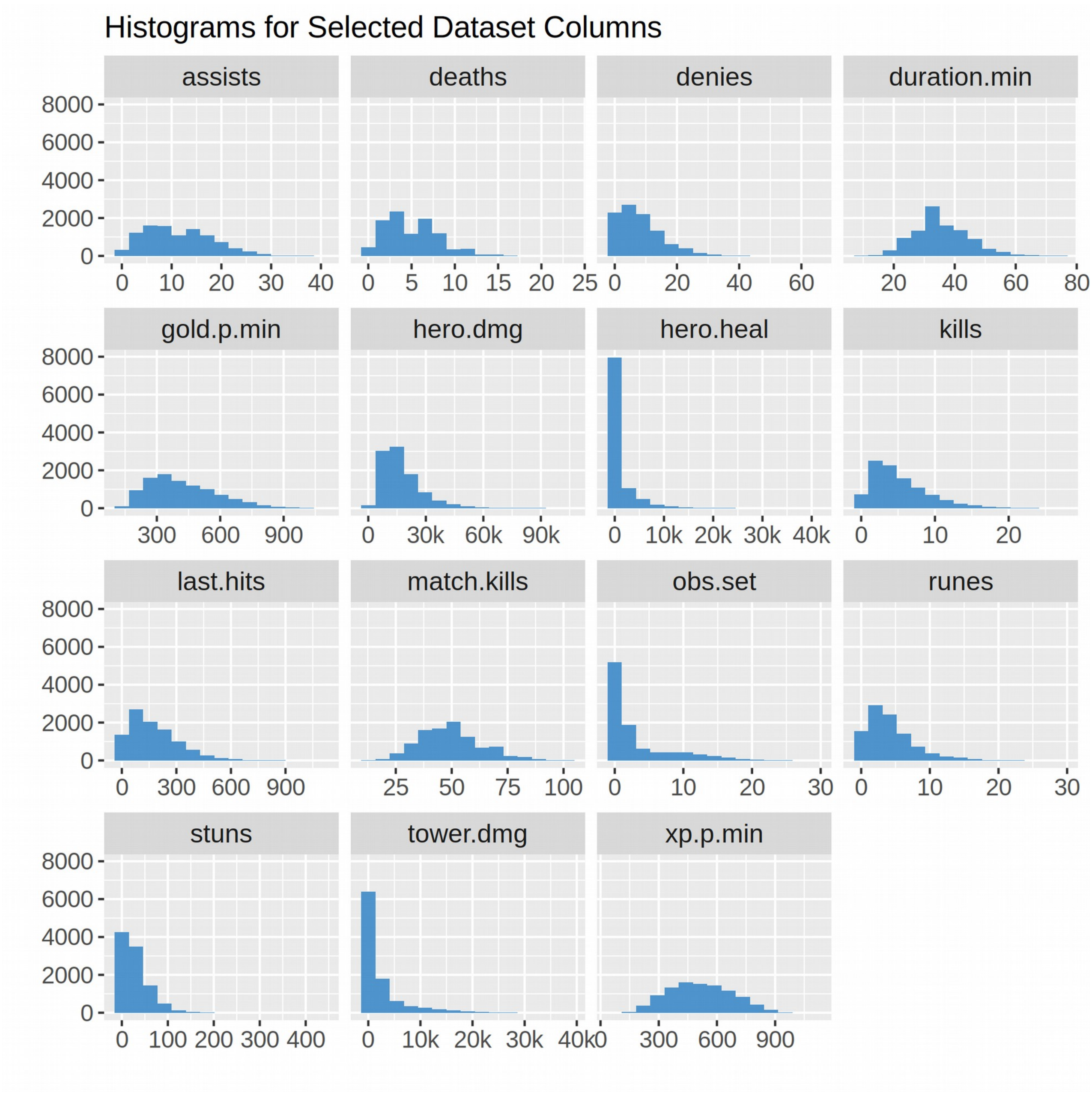


Figure 2: Histograms for many dataset columns

Boxplots for a few columns with similar ranges

Boxplots – like summary statistics and histograms – show the distribution of values for a given column, but from a slightly different perspective. Below in Figure 3 we can see the distribution for an individual player's *assists, deaths*, *denies* (minion denies), *kills* (hero kills), and *runes*. Each is on a very similar scale ranging from 0-65 with most values around 5-10. While both boxplots and histograms show a column's distribution, boxplots excel at highlighting where the majority of the points lie (the middle 50%) and if there are outliers above or below this middle set (easily distinguishable by dots past the whiskers). We can see quite a few outliers for each column highlighted below. While we already looked at the histogram visual, this boxplot also makes it easy to see that each of these 5 columns have a right skew to them.

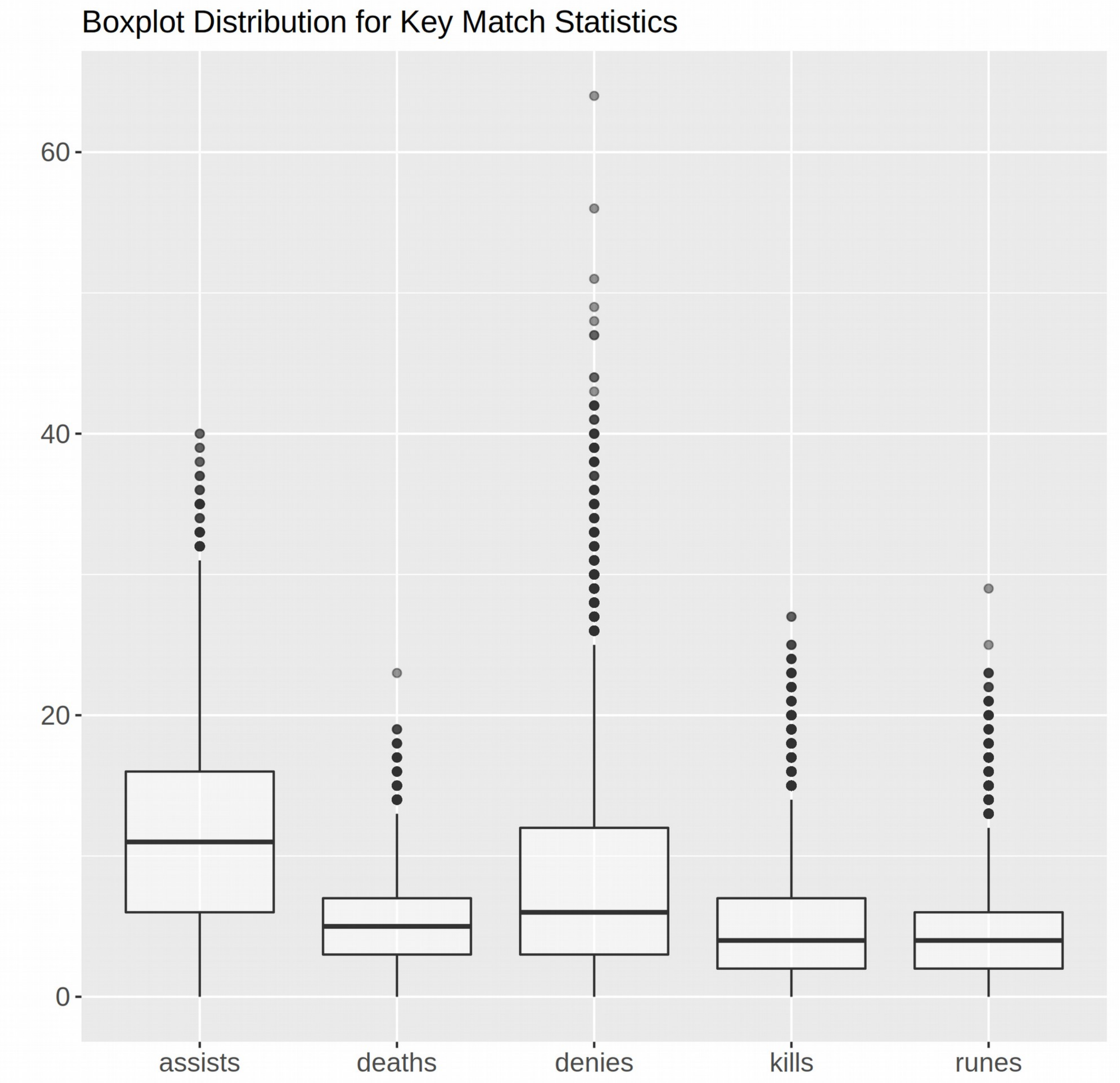


Figure 3: Boxplots for a few columns with similar value ranges

Correlation Matrix

After we have explored the dataset some and determined if there are any inaccuracies we need to investigate (missing data, column range out of line with expectations, etc), we can move to identifying relationships among the variables. Figure 4 shows the correlation matrix among many variables and indicates a strong correlation with a darker color (blue or red) while indicating the direction by the color – blue being positive and red being negative. The strongest relationship (0.88) appears to be between a player’s gold and xp per minute, which intuitively does make sense; if a player is getting a lot of gold and kills and not dying much, they will naturally level up faster than other players. We can see kills and deaths have a moderate relationship to gold and xp per minute, though in opposite directions. If you have never played a game like Dota 2 before, you can still gain a small understanding of good and bad things to happen in a match, like deaths has a negative (red) correlation with many other columns so it is probably bad, while kills and assists have mainly positive ones. From this introductory analysis, assists (damaging enemy heroes while teammate gets killing blow) seem to correlate most strongly with winning a match.

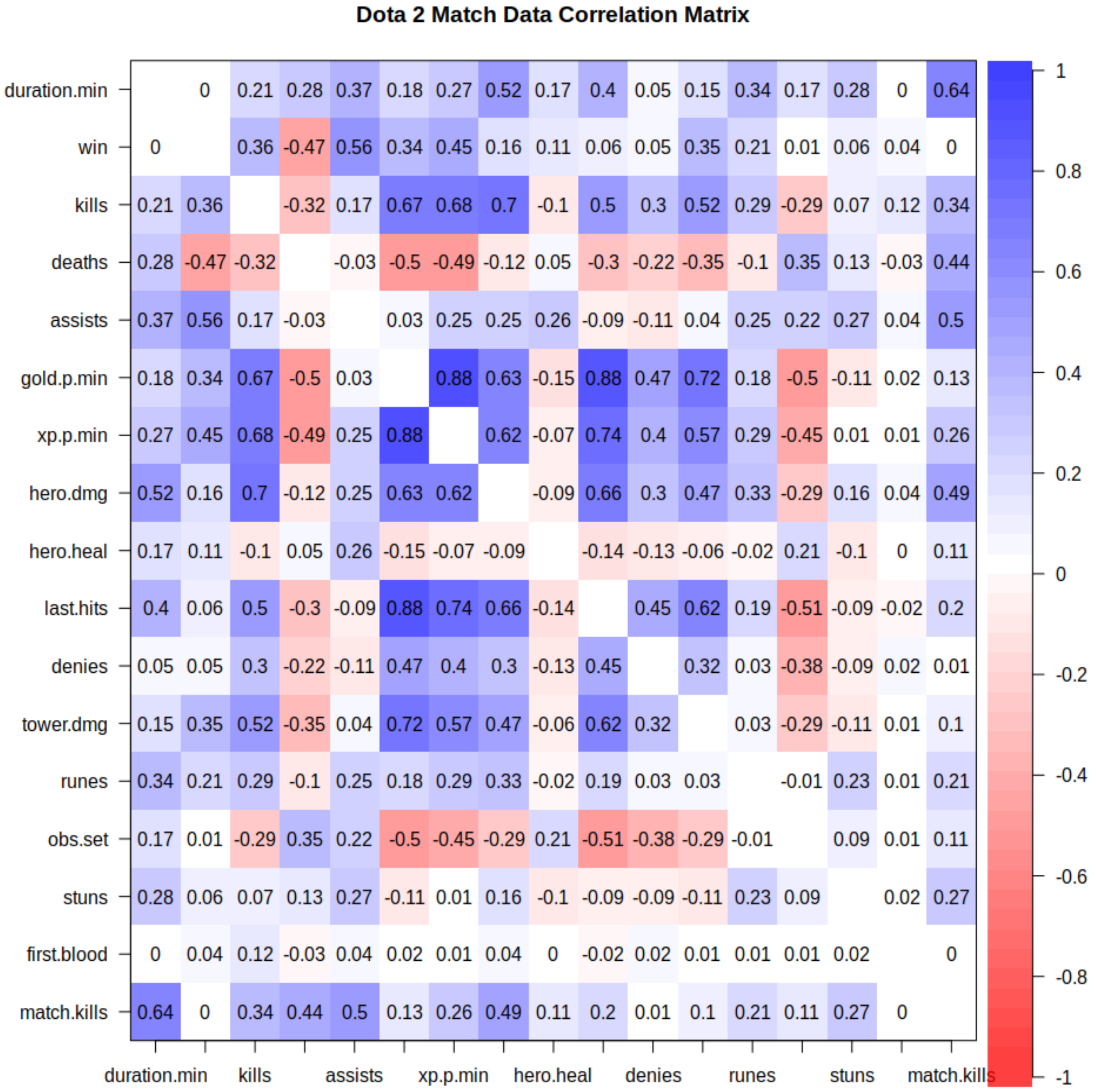


Figure 4: Correlation matrix of selected numeric columns

SQL Schema for Dataset

A SQL schema defines the columns of a dataset along with their data types, type lengths (if applicable), primary keys, and not null designations. Table 2 below defines the SQL schema for the dataset for this project. While I don’t use the field *match.id.slot* in any of my data analysis, it is the key that makes every row unique and was useful in my Python program to update the *first.blood* column with every player that scored a first blood kill with a 1 (after init every row to 0). I typically use an underscore in my Python program column names, but I am doing my exploratory analyses and visualizations in R where I like using periods. That is why my cleaned dataset has periods in some column names.

Field.Name	**Data.Type**	**Length**	**Primary.Key**	**Not.Null**
match.id	integer	10		Yes
match.id.slot	varchar	14	Yes	Yes
duration.min	decimal	2,2		
radiant.win	boolean			
isRadiant	boolean			
num.players	integer	2		
player.slot	integer	3		Yes
kills	integer	2		
deaths	integer	2		
assists	integer	2		
gold.p.min	integer	4		
xp.p.min	integer	4		
win	boolean			
hero.dmg	integer	6		
hero.heal	integer	5		
last.hits	integer	4		
denies	integer	2		
tower.dmg	integer	5		
runes	integer	2		
obs.set	integer	2		
stuns	decimal	10,7		
first.blood	boolean			
match.kills	integer	3		

Table 2: Dataset SQL Schema

SQL-based queries of dataset (Oracle SQL syntax)

- Count number of games and calculate average match duration:

```
SELECT
   COUNT(DISTINCT "MATCH.ID")  "NUM.MATCHES",
   AVG("DURATION.MIN")  "AVG.MATCH.DURATION"
FROM DOTA2_DATA;
```

- Find minimum kills, maximum kills, and average hero damage for each player slot for players that won the game:

```
SELECT DISTINCT
   "PLAYER.SLOT",
   MIN(KILLS)  "MIN.KILLS",
   MAX(KILLS)  "MAX.KILLS",
   AVG("HERO.DMG")  "AVG.HERO.DMG"
FROM DOTA2_DATA
WHERE WIN = TRUE
GROUP BY "PLAYER.SLOT";
```

- Find the match ids, total last hits, total denies for all matches for the Radiant teams that got first blood (first kill in the match):

```
SELECT DISTINCT
   "MATCH.ID",
   SUM("LAST.HITS")  "MATCH.LAST.HITS"
   SUM("DENIES")  "MATCH.DENIES",
FROM DOTA2_DATA
WHERE "ISRADIANT" = TRUE
AND "FIRST.BLOOD" = TRUE;
```

- Find the average gold per minute, xp per minute, and total hero damage for all matches with at least 25 total kills and at least 1 player with at least 500 gold per minute who picked up 2-5 runes:

```
SELECT DISTINCT
   AVG("GOLD.P.MIN")  "AVG.GPM",
   AVG("XP.P.MIN")  "AVG.XPM"
   SUM("HERO.DMG")  "TOTAL.HERO.DMG"
FROM DOTA2_DATA  TA1
WHERE "MATCH.KILLS" >= 25
AND "MATCH.ID" IN (
   SELECT "MATCH.ID"
   FROM DOTA2_DATA  TA2
   WHERE "GOLD.P.MIN" >= 500
   AND "RUNES" BETWEEN 2 AND 5);
```

II. Interpret the Results and Draw Conclusions

Q1. What is the mean number of kills in a Dota 2 match stratified by match duration?

Overall, the mean kills per match is 49.65, almost 50. Once we stratify by match duration, the story is a bit more clear – the mean kills per match increases with increasing match duration. Logically, this makes sense as a longer game will generally feature more kills as there is more time to kill the enemy characters. Interestingly, the highest duration category featured fewer kills and bucked the positive trend, though only 2 matches were in that highest category. The number of matches for each duration category is at the base of each bar; we can tell that the majority of matches were of length 20-50 minutes long with about 50 kills each. Only 3 times did the matches last less than 10 minutes or greater than 69 minutes.

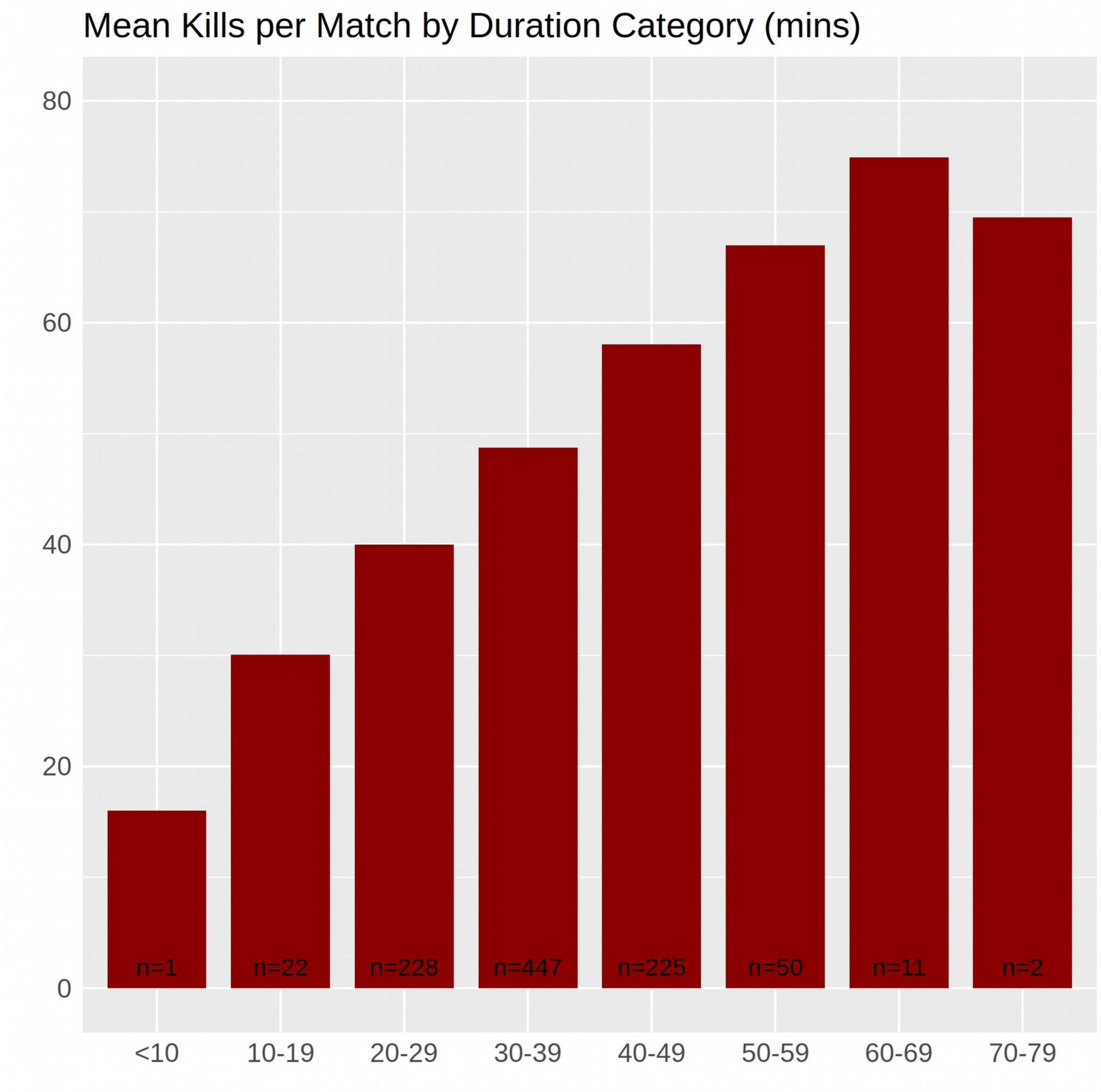


Figure 5: Q1: Bar Plot of Mean Kills per Match Split by Duration Bin

Q2. What is the distribution of match durations?

From the last question, we got a little idea of the match duration distribution, but here we can look more closely at it. Figure 6 below shows the distribution of match durations; we see it is close to being normally distributed with a median of 34.3. The tails are fairly tight meaning the standard deviation is small relative to the mean. We can tell that the vast majority of these professional Dota 2 matches last between 20 and 60 minutes long.

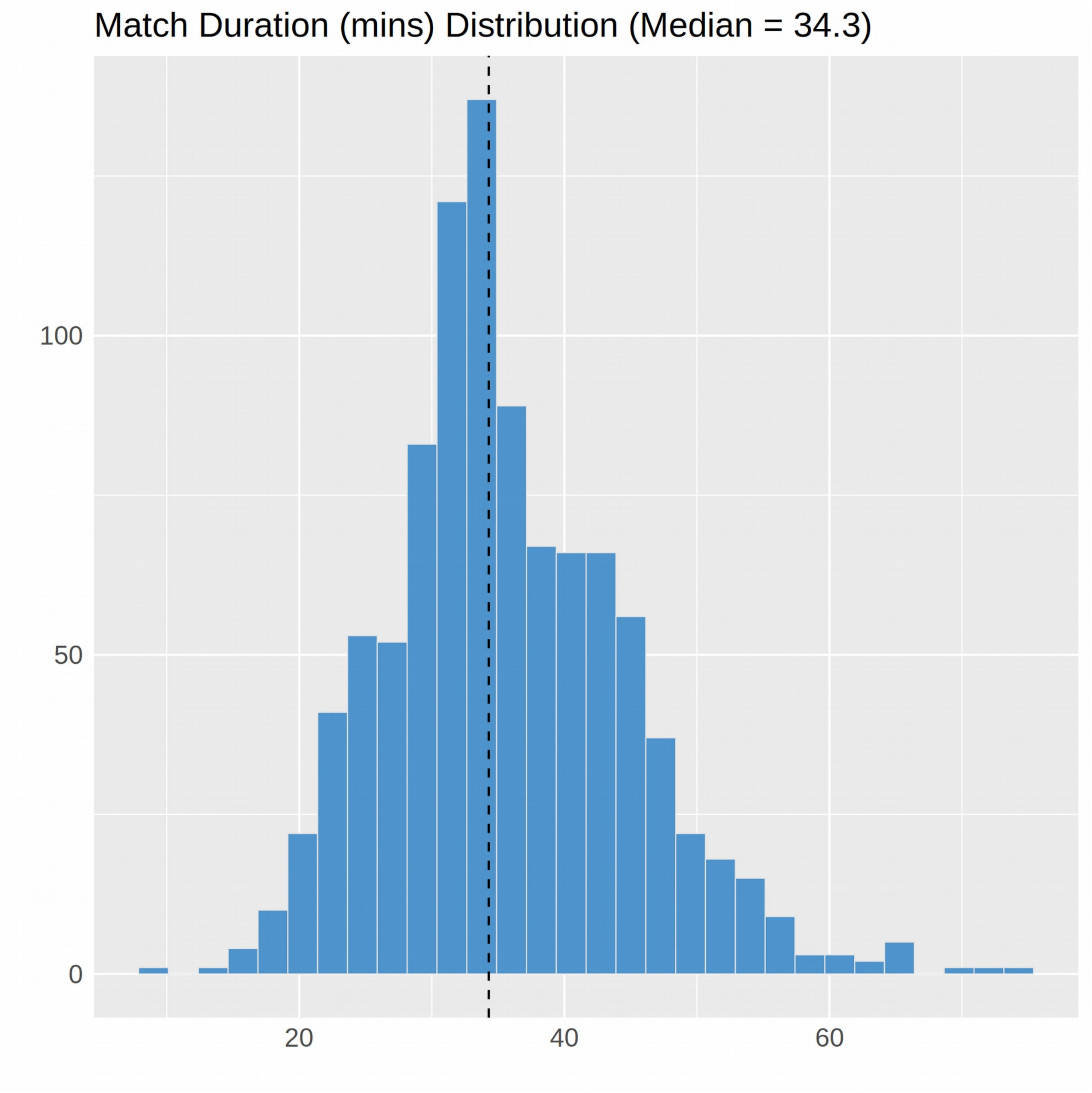


Figure 6: Histogram of Match Duration (mins)

Q3. How does kill death assist ratio (KDA) correlate with experience per minute (XPM)?

The KDA is calculated using a piecewise function (to handle 0 deaths case):

KDA = { if deaths > 0: (kills + assists) / deaths,
else: kills + assists }

I first calculated the correlation coefficient between these 2 variables and the result is 0.5276, meaning the R^2 value is 0.278. This indicates there is a weak to moderately strong relationship between a player's KDA and their experience gained per minute (XPM). Figure 7 below shows this weak relationship as a scatter plot. Since there are about 1000 matches represented on the plot, an alpha transparency value was applied to the scatter plot to better show each individual point and clusters of points. The blue line is a linear regression line added with R's *geom_smooth* function with the 'lm' parameter to force a linear model to be used instead of 'loess' smoothing.

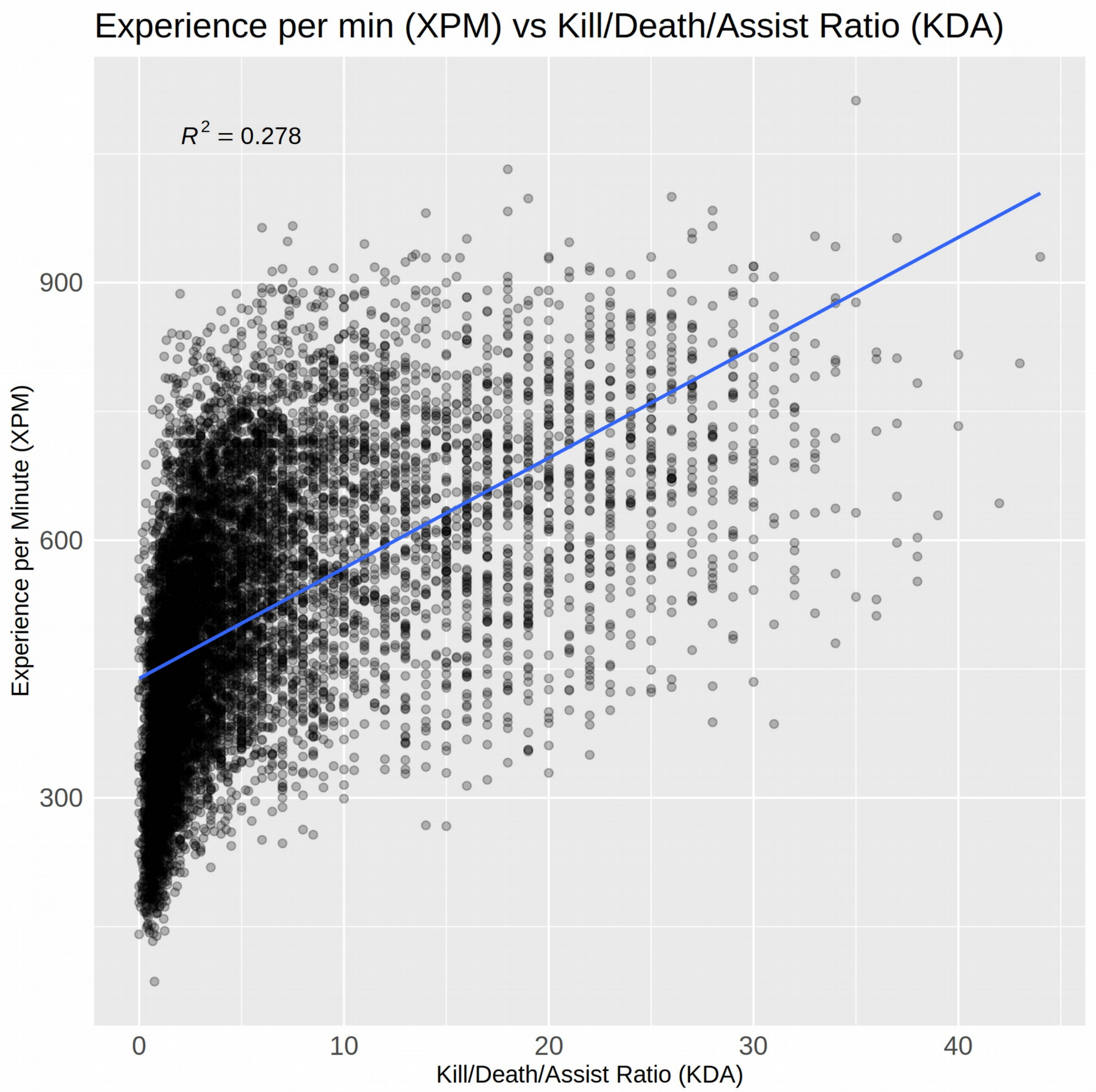


Figure 7: Scatter Plot of XPM vs KDA

Q4. Does the team that drew first blood (got the first game kill) win more often?

I had a feeling that teams that drew first blood (scored the first enemy character kill) would win more often, but 57% is pretty good.   This seems to indicate that scoring the first blood kill may lead to a snowball effect where the team gets a small advantage and continually builds upon it culminating in a victory. Honestly, for a long time my program calculated a 96% win rate for teams with first blood kills, until I reviewed my Python parsing program and corrected a bug where a first blood kill was defined by finding the *last* kill (max kill time in a log of kills and times) instead of the *first* kill (min kill time). It definitely makes sense that the wrong version indicated a 96% win rate as a winning team will be stronger and very often gets the final kill in a match to destroy all the towers and the enemy's ancient to win the match. These corrected numbers seem a lot more reasonable and it should have been a red flag that the results were almost too strong, too good to be true.

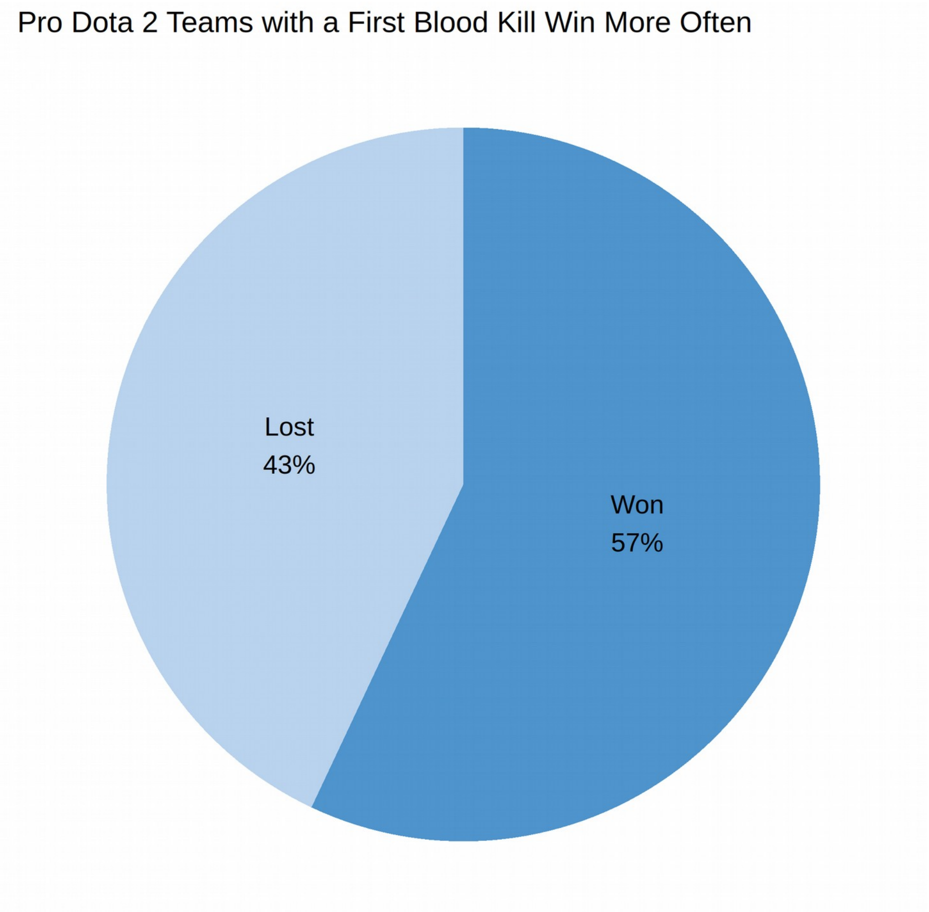


Figure 8: Pie chart of Win % for Teams Who Drew First Blood

Q5. Does gold per minute (GPM) or experience per minute (XPM) have a higher correlation with a match win?

I looked at the correlation coefficient R and R^2 value for both cases – Win vs GPM had a correlation coefficient R of 0.342 and R^2 of 0.1169, while Win vs XPM had a higher correlation (R of 0.4471 and R^2 of 0.1999). Neither one of these relationships are very strong, though. This could be due to the fact that GPM, XPM, and a Win are all after the fact indicators aggregating decision making during a match; none of them can tell a player where to improve, only that accumulating more gold and experience per minute is good and leads to wins.

What I found interesting in trying to answer this question was the visual of gold per minute and xp per minute for winning and losing teams. Figure 9 below shows that winning teams have higher GPMs and XPMs relative to losing teams, as we would expect.

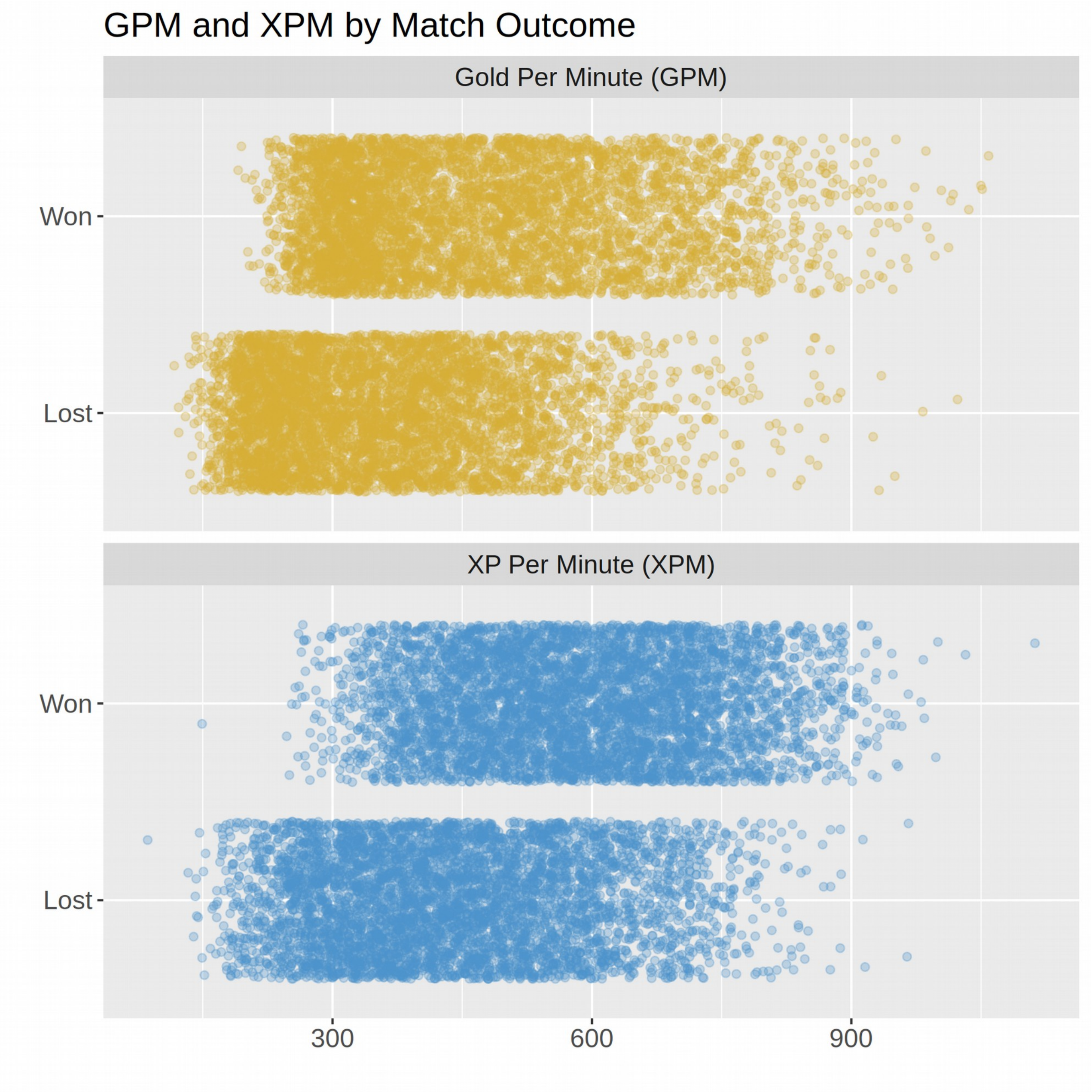
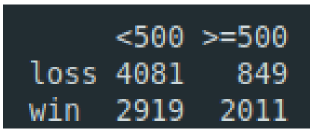


Figure 9: Jittered Scatter Plot of GPM and XPM by Win/Lost

Q6. Do players with 500 or more GPM win matches more often than players with less than 500 GPM?

Though the relationship between a players having greater than or equal to 500 GPM and winning a game is quite weak (R^2 of 0.067), it does seem like (from Table 3) players with >= 500 GPM do win more their matches more often. Since this analysis is at the player level and not all players on a team will reach 500 GPM but still win, a better question to answer is do *winning teams* have a significant difference in median team GPM? We can investigate that in Q7 below.



```
     <500 >=500
loss 4081   849
win  2919  2011
```

Table 3: GPM vs Win

Q7. Do winning teams have a significant difference in median GPM?

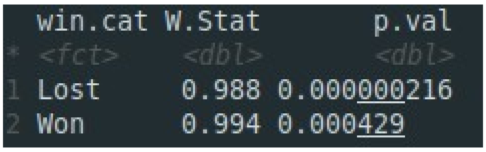
H_o (null) : there is no significant difference in median GPM between winning and losing teams

Median GPM_{win} = Median GPM_{lose}

H_A (alternative) : there is a significant difference in mean GPM between winning and losing teams

Median $GPM_{win} \neq$ Median GPM_{lose}

Testing Assumptions of Student's T Test:
1. Normality with Shapiro-Wilk test:
Null hypothesis = groups are normally distributed
Alternative hypothesis = groups are not normally distributed



```
  win.cat W.Stat       p.val
* <fct>    <dbl>       <dbl>
1 Lost     0.988 0.000000216
2 Won      0.994 0.000429
```

Table 4: Results of Shapiro-Wilk test

We can see in Table 4 above that both p-values are significant, so we must reject the null hypothesis and conclude that both groups are *not* normally distributed. We can visually see this in quantile-quantile (QQ) plots on the next page in Figure 10.

Normality with QQ plot of Median Team GPM – most points fall approximately across the line, however several are outside the 95% confidence interval bands for the Lost group. This confirms our previous Shapiro-Wilk test findings of normality violations.

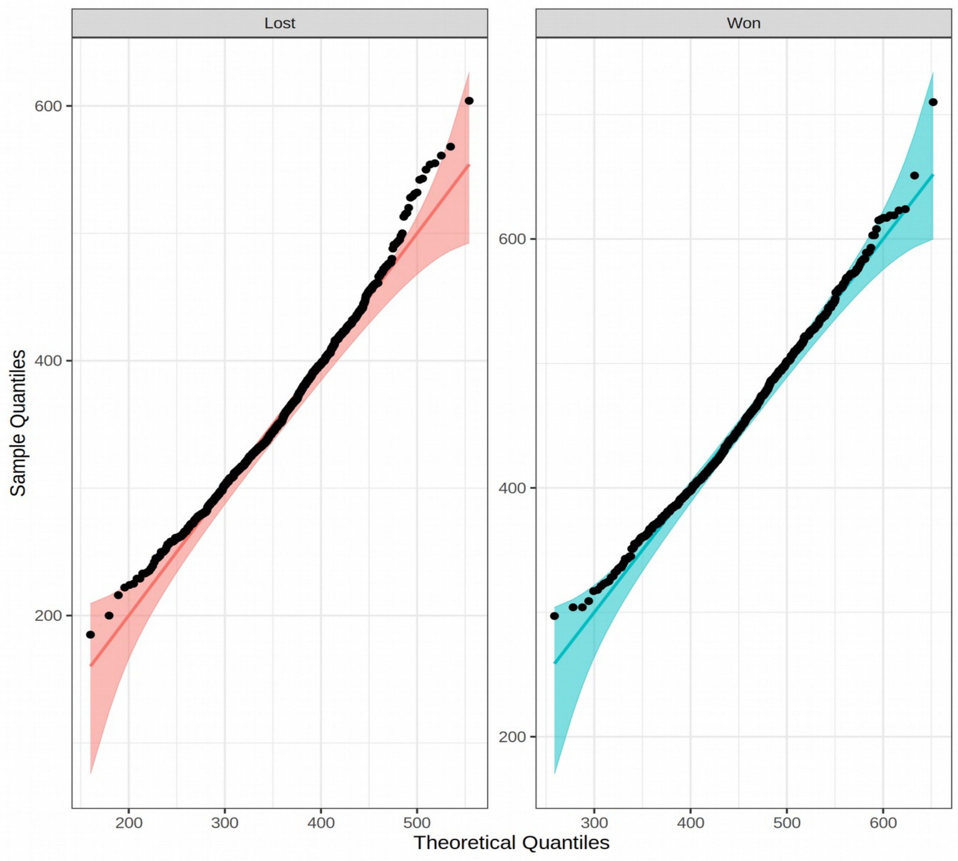


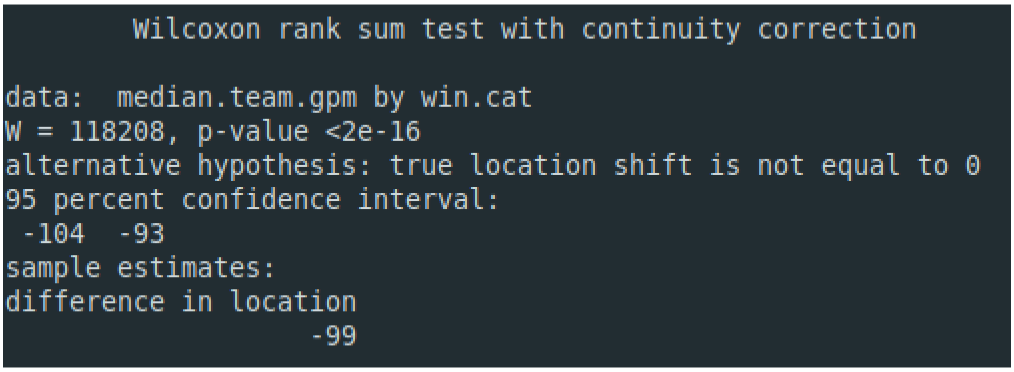
Figure 10: QQ Plot Testing Normality of Median Team GPM

Wilcoxon Rank Sum Test

Since normality is violated, a student's t-test is no longer an appropriate test to perform on this dataset. A non-parametric version is available in the Wilcoxon Rank Sum Test ("Independent Samples T-test in R | Statistical Methods", n.d.).

- Null hypothesis = equal medians between Won and Lost groups
- Alternative hypothesis = medians of the 2 populations are not equal

Figure 11 below shows the results of a Wilcoxon rank sum test for median team gpm by win category (Won, Lost). The p-value is very significant, so we reject the null hypothesis and can conclude that the medians of the Won and Lost distributions do indeed differ.



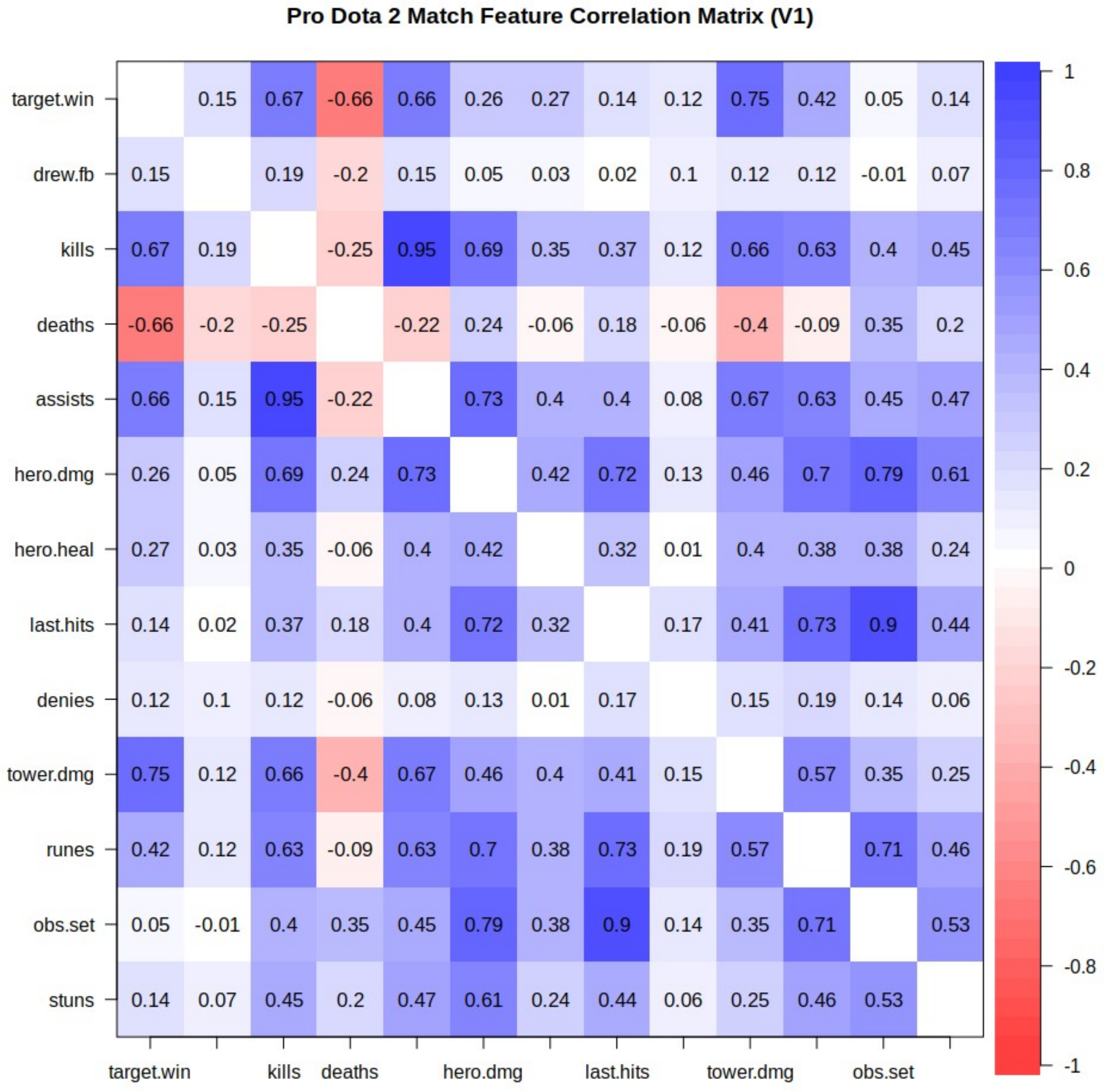
```
        Wilcoxon rank sum test with continuity correction

data:  median.team.gpm by win.cat
W = 118208, p-value <2e-16
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -104  -93
sample estimates:
difference in location
                   -99
```

Figure 11: Wilcoxon Rank Sum Test Results

Q7. What features lead to an edge in winning (> 50% accuracy)?

Exploring available features and their correlations

Data was aggregated to the team level from the individual player level to better distinguish good team play features that lead to win or are illustrative of winning teams. Figure 12 below shows an initial correlation matrix of some features to consider using in our model to predict wins. Immediately we can see a few features that highly correlated with a win – *kills, assists, tower.dmg* – and a few that are highly correlated with each other – *kills*, *assists*, *hero.dmg* – and may need to be combined to produce a better model. On the next page this matrix will be pared down to fewer features that seem to correlate most strongly with the response variable *target.win*.



Pro Dota 2 Match Feature Correlation Matrix (V1)

	target.win		kills	deaths		hero.dmg		last.hits		tower.dmg		obs.set	
target.win		0.15	0.67	-0.66	0.66	0.26	0.27	0.14	0.12	0.75	0.42	0.05	0.14
drew.fb	0.15		0.19	-0.2	0.15	0.05	0.03	0.02	0.1	0.12	0.12	-0.01	0.07
kills	0.67	0.19		-0.25	0.95	0.69	0.35	0.37	0.12	0.66	0.63	0.4	0.45
deaths	-0.66	-0.2	-0.25		-0.22	0.24	-0.06	0.18	-0.06	-0.4	-0.09	0.35	0.2
assists	0.66	0.15	0.95	-0.22		0.73	0.4	0.4	0.08	0.67	0.63	0.45	0.47
hero.dmg	0.26	0.05	0.69	0.24	0.73		0.42	0.72	0.13	0.46	0.7	0.79	0.61
hero.heal	0.27	0.03	0.35	-0.06	0.4	0.42		0.32	0.01	0.4	0.38	0.38	0.24
last.hits	0.14	0.02	0.37	0.18	0.4	0.72	0.32		0.17	0.41	0.73	0.9	0.44
denies	0.12	0.1	0.12	-0.06	0.08	0.13	0.01	0.17		0.15	0.19	0.14	0.06
tower.dmg	0.75	0.12	0.66	-0.4	0.67	0.46	0.4	0.41	0.15		0.57	0.35	0.25
runes	0.42	0.12	0.63	-0.09	0.63	0.7	0.38	0.73	0.19	0.57		0.71	0.46
obs.set	0.05	-0.01	0.4	0.35	0.45	0.79	0.38	0.9	0.14	0.35	0.71		0.53
stuns	0.14	0.07	0.45	0.2	0.47	0.61	0.24	0.44	0.06	0.25	0.46	0.53	

Figure 12: Initial Correlation Matrix of Dataset Features

Selecting good model features

Version 2 of the correlation matrix is shown below in Figure 13 after we have thrown out *kills* in favor of *assists*, removed low correlation features – *drew.fb, hero.dmg, hero.heal, last.hits, denies, obs.set*, and *stuns*. Also, the *tower damage* column was divided by a 1,000 to bring its scale in line with other columns.

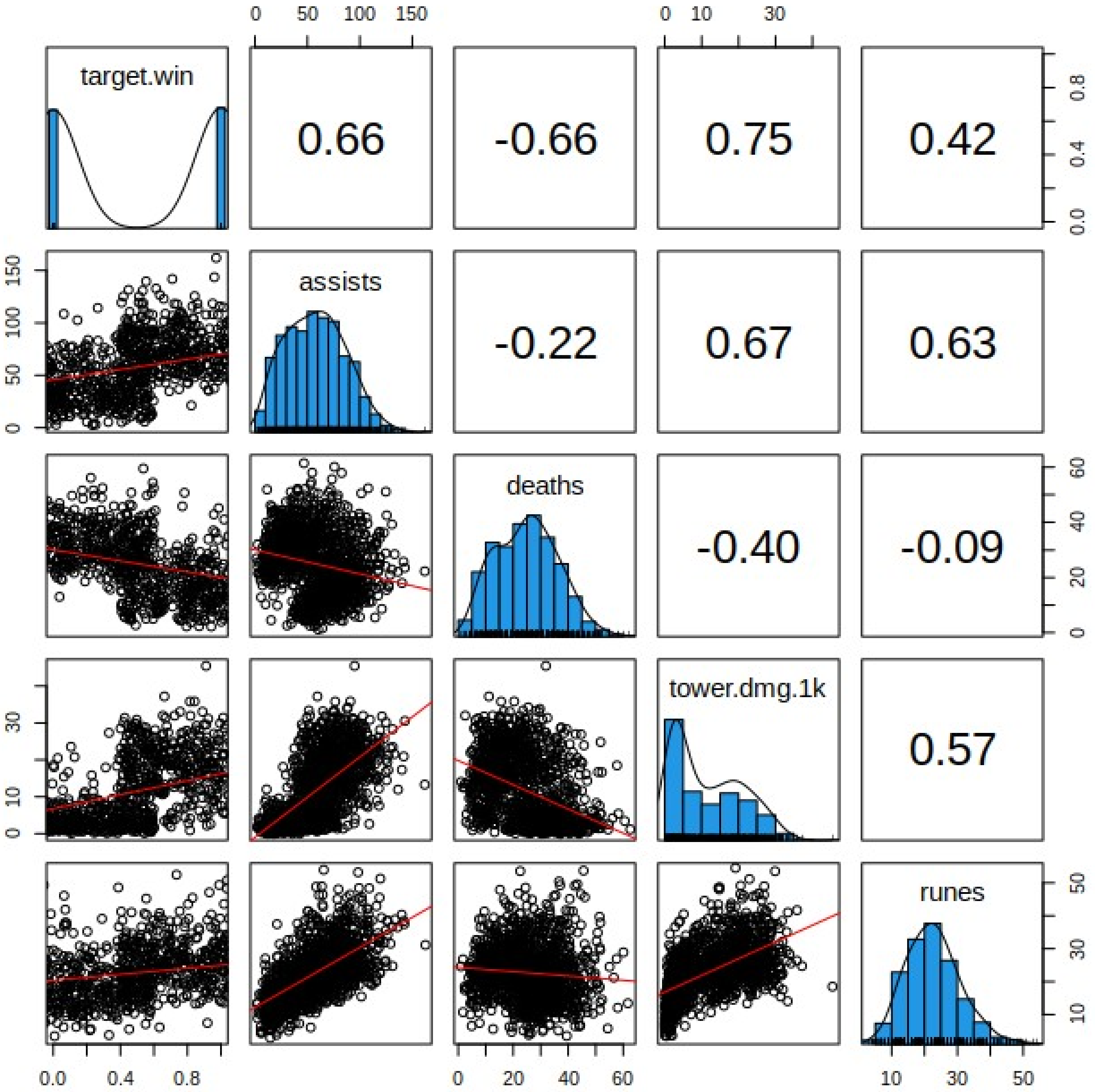
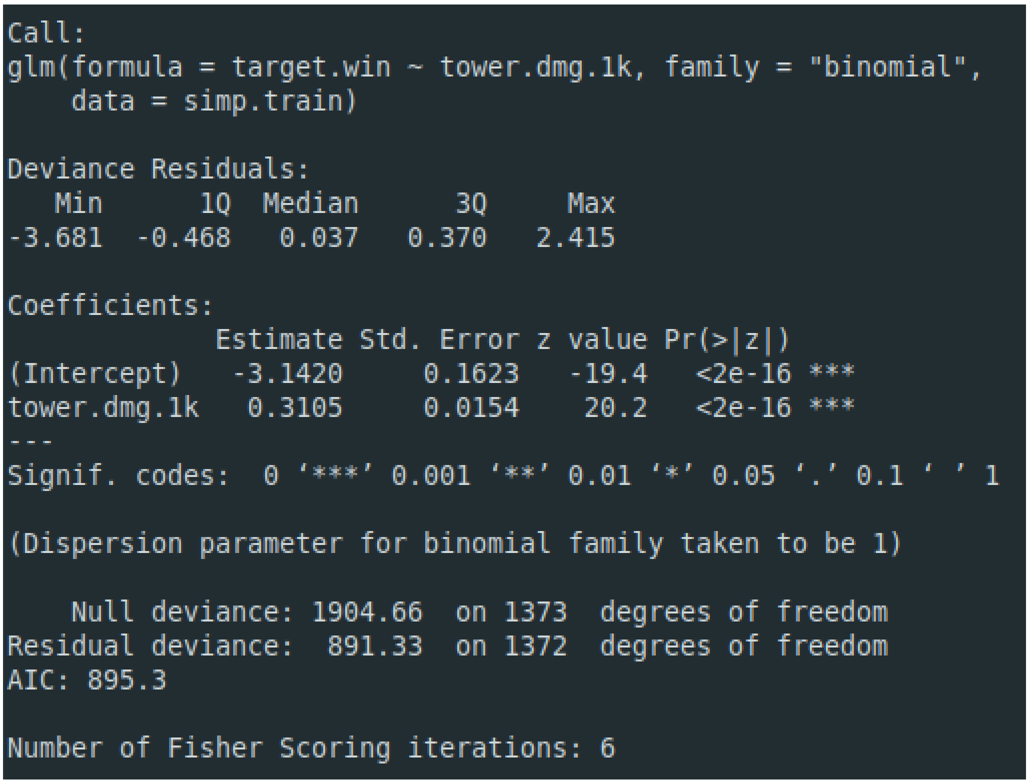


Figure 13: Pairs Plot after Narrowing Feature Set

Fitting a logistic regression to predict a match win

First, we will try a univariate logistic regression model only using the predictor *tower damage 1k.* Figure 14 below shows the univariate model results – an Akaike Information Criterion (AIC) of 895.3 and a 95% confidence interval (CI) for *tower damage 1k* of 1.325 to 1.407.

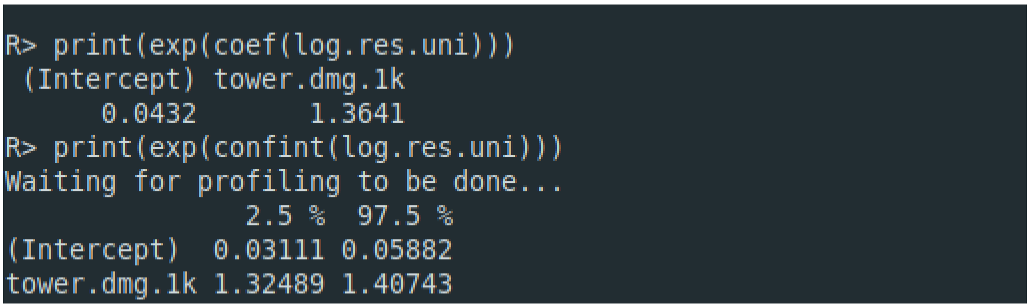


```
Call:
glm(formula = target.win ~ tower.dmg.1k, family = "binomial",
    data = simp.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-3.681  -0.468   0.037   0.370   2.415

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.1420     0.1623   -19.4   <2e-16 ***
tower.dmg.1k   0.3105     0.0154    20.2   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1904.66  on 1373  degrees of freedom
Residual deviance:  891.33  on 1372  degrees of freedom
AIC: 895.3

Number of Fisher Scoring iterations: 6
```

Figure 14: Univariate Model Results with Predictor Tower Damage 1k

By exponentiating the model coefficients, we can talk about the predictor influence on the response variable. For this univariate model, we can say that the odds of winning a match increase by about 36% for every 1,000 units of tower damage done by a team. This is calculated using Figure 15:

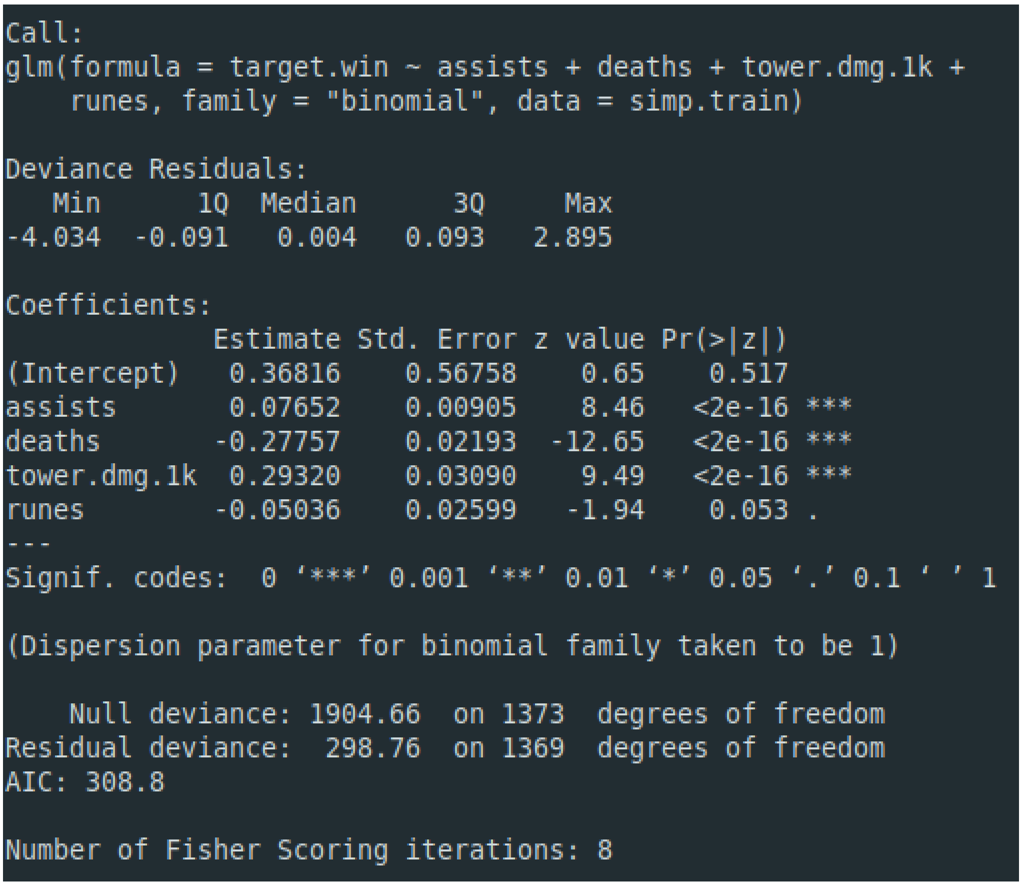
% (in/de)crease in response per unit of predictor = percent(coef – 1)
= percent(1.3641 – 1) = 36.41%



```
R> print(exp(coef(log.res.uni)))
 (Intercept) tower.dmg.1k
      0.0432       1.3641
R> print(exp(confint(log.res.uni)))
Waiting for profiling to be done...
               2.5 %  97.5 %
(Intercept)  0.03111 0.05882
tower.dmg.1k 1.32489 1.40743
```

Figure 15: Model Interpretation by Exponentiating Coefficients and CI

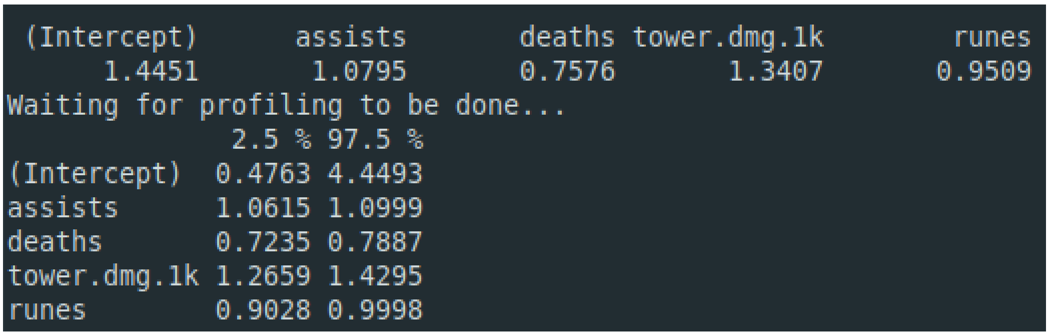
After creating a multivariate model on the next page, we can compare the model results – the AIC relative quality of the model, power of the model looking at CI widths, and the model goodness of fit with Hosmer-Lemeshow fit tests (Koh, 2020).



```
Call:
glm(formula = target.win ~ assists + deaths + tower.dmg.1k +
    runes, family = "binomial", data = simp.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-4.034  -0.091   0.004   0.093   2.895

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.36816    0.56758    0.65    0.517
assists       0.07652    0.00905    8.46   <2e-16 ***
deaths       -0.27757    0.02193  -12.65   <2e-16 ***
tower.dmg.1k  0.29320    0.03090    9.49   <2e-16 ***
runes        -0.05036    0.02599   -1.94    0.053 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1904.66  on 1373  degrees of freedom
Residual deviance:  298.76  on 1369  degrees of freedom
AIC: 308.8

Number of Fisher Scoring iterations: 8
```

Figure 16: First Logistic Regression Model Results

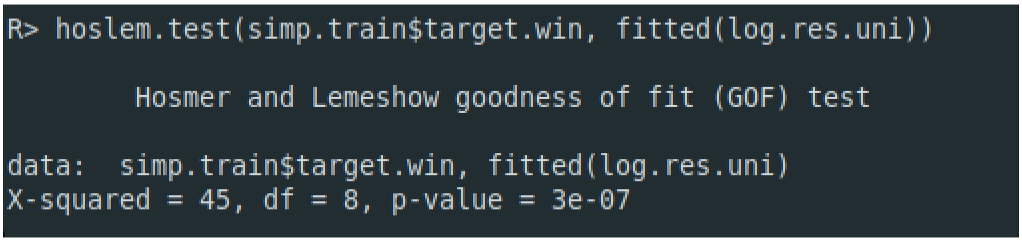
Now a logistic regression model is fit to this new dataset and the results are shown above in Figure 16. *Assists*, *deaths*, and *tower damage 1k* are all very significant indicators of a match win, while *runes* is barely significant at the 95% confidence level. Figure 17 below gives us a better idea of what this model is saying. If we focus on *deaths* for instance, we can conclude that after adjusting for all the confounders (all model features besides *deaths*), the odds of winning a match decrease by about 24% (0.7576 - 1 as a percent) for every team death. *Runes* also decrease the odds of winning by 5% for every rune picked up, after adjusting for all the confounders besides *runes*. This is unexpected and surprising as these runes can be quite powerful when used during a match. The 2 positive relationships lie with *assists* and *tower damage 1k* that increase the odds of winning (after adjust for confounders) by 8% per assist and 34% per 1000 tower damage, respectively.



```
 (Intercept)      assists       deaths tower.dmg.1k        runes
      1.4451       1.0795       0.7576       1.3407       0.9509
Waiting for profiling to be done...
              2.5 % 97.5 %
(Intercept)  0.4763 4.4493
assists      1.0615 1.0999
deaths       0.7235 0.7887
tower.dmg.1k 1.2659 1.4295
runes        0.9028 0.9998
```

Figure 17: Model Interpretation by Exponentiating Coefficients and CI

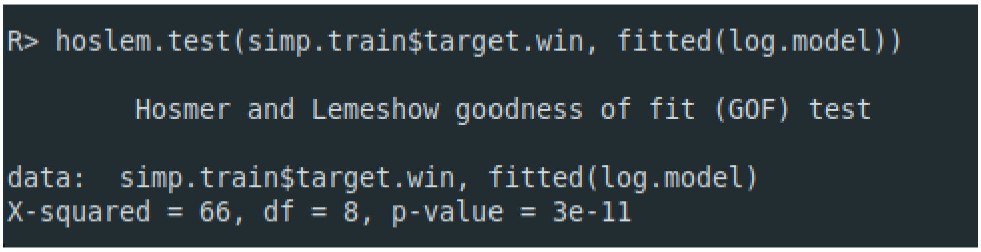
Comparing to our first model (univariate with *tower damage 1k* predictor variable), this multivariate model seems to be a better model. It has a lower AIC (308.8 vs 893.5) indicating a higher relative model quality, but with an increased number of predictors the new model has widened the 95% CI for *tower damage* 1k from 1.325 to 1.407 to now 1.2659 to 1.4295 (Koh, 2020). Lastly, we can compare the goodness of fit for each model with a Hosmer-Lemeshow Goodness of fit test.



```
R> hoslem.test(simp.train$target.win, fitted(log.res.uni))

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  simp.train$target.win, fitted(log.res.uni)
X-squared = 45, df = 8, p-value = 3e-07
```

Figure 18: Univariate Model Goodness of fit Test Results

Figure 18 above shows the univariate model results from a Hosmer-Lemeshow Goodness of fit test. Since the p-value is very significant (3e-07 is quite small), we *reject* the null hypothesis and conclude that there *is* significant evidence to show that the model is a poor fit to the data (Koh, 2020). For the multivariate model, the results in Figure 19 below indicate that it also has significant evidence to show that the model is a poor fit to the data.



```
R> hoslem.test(simp.train$target.win, fitted(log.model))

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  simp.train$target.win, fitted(log.model)
X-squared = 66, df = 8, p-value = 3e-11
```

Figure 19: Multivariate Model Goodness of fit Test Results

Using multivariate model in predictions

- Probability of winning a match with 4 assists, 0 deaths, 5000 tower damage, and 1 rune:

$$\log\left(\frac{p}{1-p}\right)=0.36816+0.07652\times Assists-0.27757\times Deaths+0.29320\times TowerDmg\,1k-0.05036\times Runes$$

$$\log\left(\frac{p}{1-p}\right)=0.36816+0.07652\times 4-0.27757\times 0+0.29320\times 5-0.05036\times 1$$

$$\log\left(\frac{p}{1-p}\right)=2.09$$

$$\frac{p}{1-p}=\exp(2.09)=8.085$$ (odds ratio – 8:1 odds of winning match)

p = 0.8899 = 88.99% ~ **89%** (probability of win)

- Probability of winning a match with 10 assists, 11 deaths, 7234 tower damage, and 4 rune:

$$\log\left(\frac{p}{1-p}\right)=0.36816+0.07652\times Assists-0.27757\times Deaths+0.29320\times TowerDmg\,1k-0.05036\times Runes$$

$$\log\left(\frac{p}{1-p}\right)=0.36816+0.07652\times 10-0.27757\times 11+0.29320\times 7.234-0.05036\times 4$$

$$\log\left(\frac{p}{1-p}\right)=-0.0003615$$

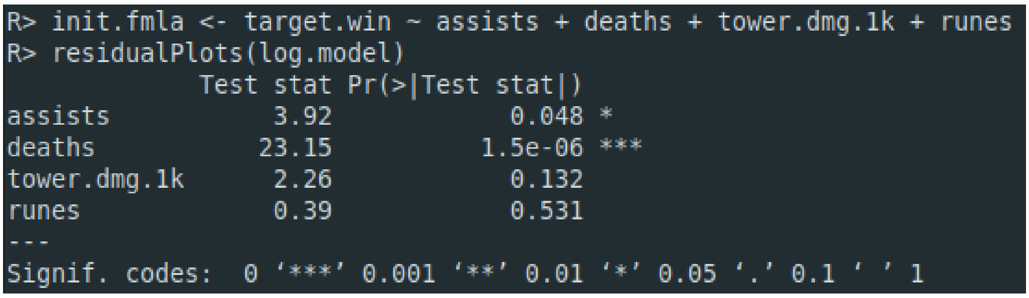
$$\frac{p}{1-p}=\exp(-0.0003615)=0.9996$$ (odds ratio – 1:1 odds of winning match)

p = 0.4999 = 49.99% ~ **50%** (probability of win)

Residual plotting, testing model fit

In his journal article on residuals and logistic regression, Zhang (2016) recommends performing a lack-of-fit test in addition to the standard residuals against predictors or fitted values plots. Zhang points out that

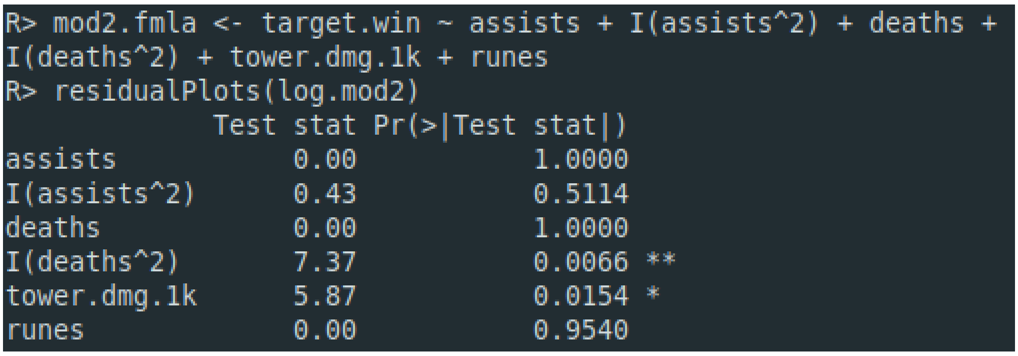
> "Visual inspection is only a rough estimation and cannot be used as a rule to modify the model, [but the R function *residualPlots()* from *car* package] performs formal statistical testing (lack-of-fit test) to see if a variable has a relationship with residuals".



```
R> init.fmla <- target.win ~ assists + deaths + tower.dmg.1k + runes
R> residualPlots(log.model)
             Test stat Pr(>|Test stat|)
assists           3.92            0.048 *
deaths           23.15          1.5e-06 ***
tower.dmg.1k      2.26            0.132
runes             0.39            0.531
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 20: Initial Multivariate Model Lack-of-fit Test Results

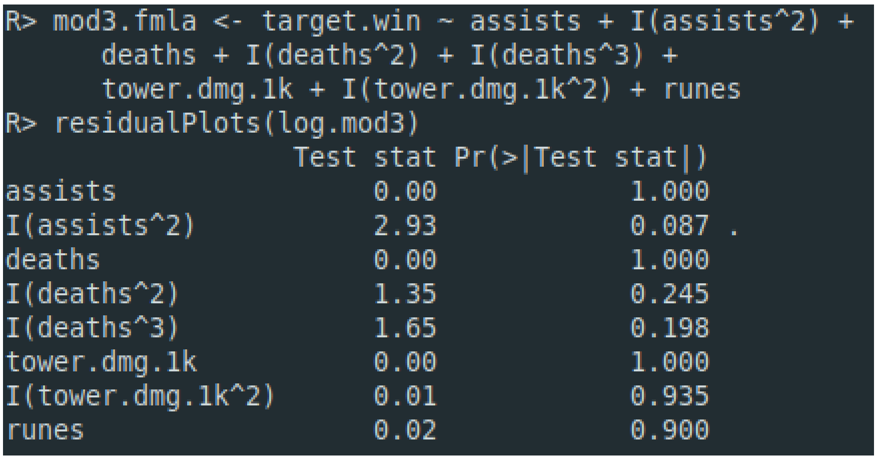
As shown in Figure 20 above, running the *residualPlots()* function on the multivariate model from above results in 2 variables (*deaths, assists)* that are statistically significant and that adding a quadratic term for each is reasonable. Figure 21 below shows the updated model with quadratic terms for both *deaths* and *assists*. We can see that now there are 2 more variables that the fit test suggests adding squared terms for – *deaths*^{2} and *tower.dmg.1k*.



```
R> mod2.fmla <- target.win ~ assists + I(assists^2) + deaths +
I(deaths^2) + tower.dmg.1k + runes
R> residualPlots(log.mod2)
             Test stat Pr(>|Test stat|)
assists           0.00           1.0000
I(assists^2)      0.43           0.5114
deaths            0.00           1.0000
I(deaths^2)       7.37           0.0066 **
tower.dmg.1k      5.87           0.0154 *
runes             0.00           0.9540
```

Figure 21: Multivariate Model V2 After Adding 2 Squared Terms

For the third iteration, *deaths*^{3} and *tower.dmg.1k*^{2} are added to the model formula. Figure 22 below shows the results and finally it says that the model is properly specified.

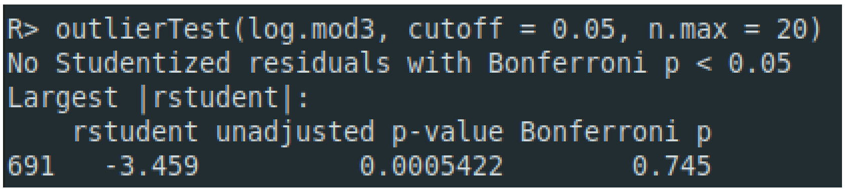


```
R> mod3.fmla <- target.win ~ assists + I(assists^2) +
      deaths + I(deaths^2) + I(deaths^3) +
      tower.dmg.1k + I(tower.dmg.1k^2) + runes
R> residualPlots(log.mod3)
                  Test stat Pr(>|Test stat|)
assists                0.00            1.000
I(assists^2)           2.93            0.087 .
deaths                 0.00            1.000
I(deaths^2)            1.35            0.245
I(deaths^3)            1.65            0.198
tower.dmg.1k           0.00            1.000
I(tower.dmg.1k^2)      0.01            0.935
runes                  0.02            0.900
```

Figure 22: Multivariate Model V3 After Adding 2 More Terms

Outliers testing using studentized residuals

We can also test the model for outliers at 95% confidence level. Outliers due to their extreme values can have significant impacts on model fitting (Zhang, 2016).

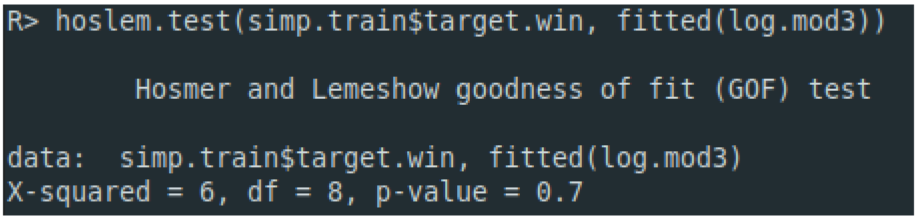


```
R> outlierTest(log.mod3, cutoff = 0.05, n.max = 20)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
    rstudent unadjusted p-value Bonferroni p
691   -3.459          0.0005422        0.745
```

Figure 23: Outlier Testing using Car Package

Goodness of fit testing

Previously our first 2 models had significant evidence to show that the models were a poor fit for the data, and now our latest model test results in Figure 24 below no longer have a significant p-value. This means we *fail to reject* the null hypothesis and conclude that there is *no* significant evidence to show that this model is a poor fit for the data. This is wonderful news!

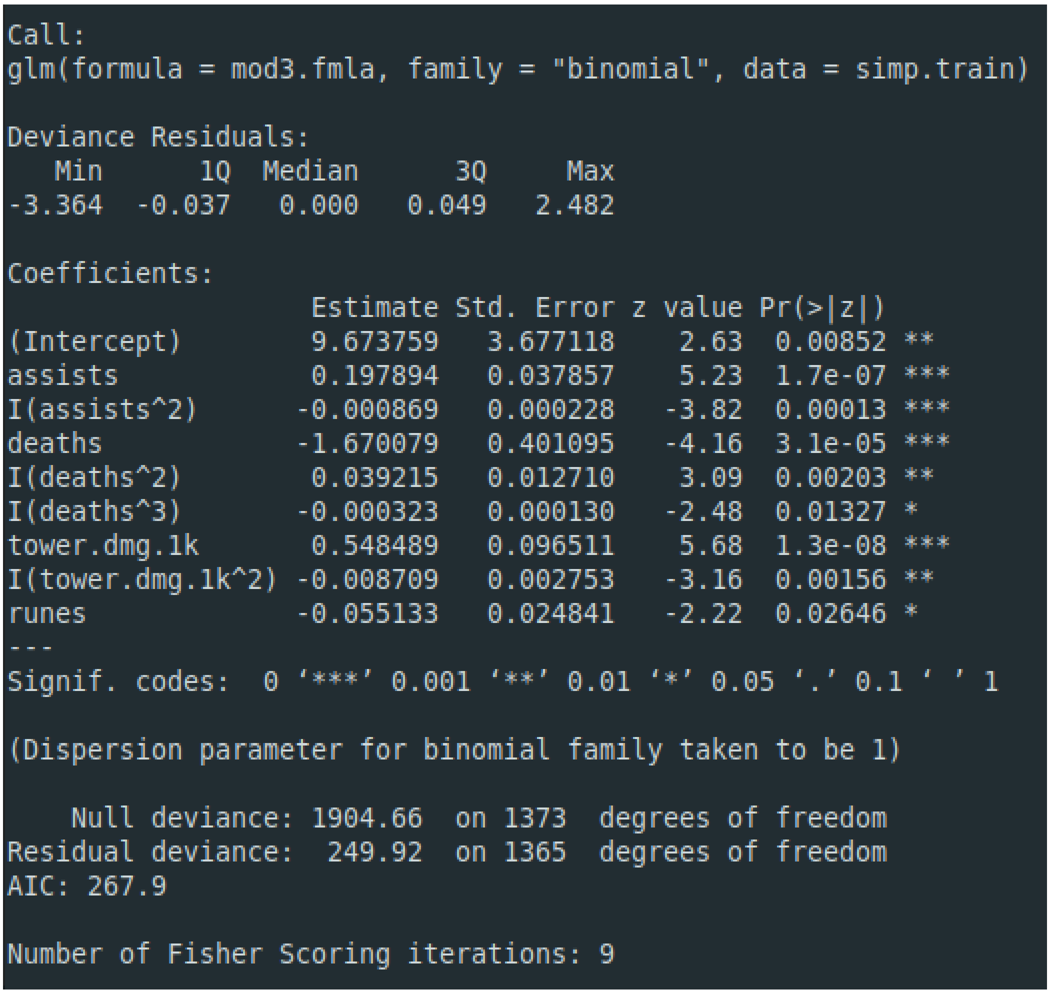


```
R> hoslem.test(simp.train$target.win, fitted(log.mod3))

        Hosmer and Lemeshow goodness of fit (GOF) test

data:  simp.train$target.win, fitted(log.mod3)
X-squared = 6, df = 8, p-value = 0.7
```

Figure 24: Hosmer-Lemeshow Goodness of Fit Test Results

Final model results

Now let’s take a look at the actual model summary for this last iteration. Figure 25 below shows that our AIC is now 267.9 – down from our initial multivariate model’s AIC of 308.8. We have added 4 more terms to our model formula and made it more complex (a total of 8 terms now).



```
Call:
glm(formula = mod3.fmla, family = "binomial", data = simp.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-3.364  -0.037   0.000   0.049   2.482

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        9.673759   3.677118    2.63  0.00852 **
assists            0.197894   0.037857    5.23  1.7e-07 ***
I(assists^2)      -0.000869   0.000228   -3.82  0.00013 ***
deaths            -1.670079   0.401095   -4.16  3.1e-05 ***
I(deaths^2)        0.039215   0.012710    3.09  0.00203 **
I(deaths^3)       -0.000323   0.000130   -2.48  0.01327 *
tower.dmg.1k       0.548489   0.096511    5.68  1.3e-08 ***
I(tower.dmg.1k^2) -0.008709   0.002753   -3.16  0.00156 **
runes             -0.055133   0.024841   -2.22  0.02646 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1904.66  on 1373  degrees of freedom
Residual deviance:  249.92  on 1365  degrees of freedom
AIC: 267.9

Number of Fisher Scoring iterations: 9
```

Figure 25: Multivariate Model V3 Results Summary

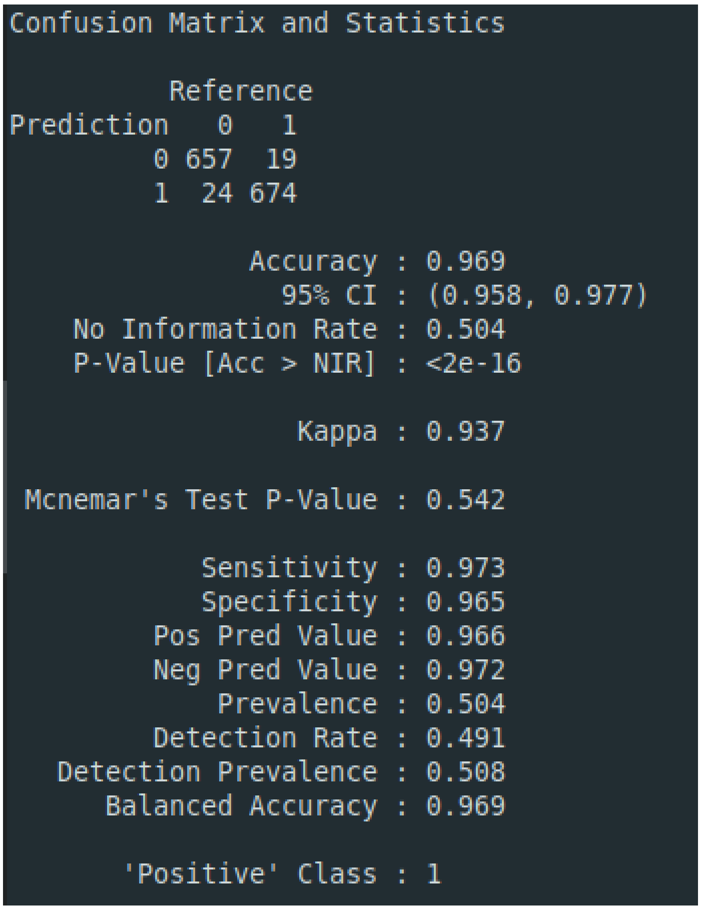
The final model coefficients are below in Table 5. We can see that each death affects the biggest change in odds of winning a match (an 81% decrease). On the positive side, the odds of winning increase by almost 22% for each assist a team gets and about 73% for each 1000 units of tower damage a team inflicts on enemy structures. From this analysis, it seems like *deaths*, *assists*, and *tower damage* are the most important factors contributing to a team winning a Pro Dota 2 match. *Kills* (discarded in favor of *assists*) would probably be important as well.

Term	exp(coefficient)	% win odds change
(Intercept)	15890	
assists	1.219	21.90%
assists^2	0.9991	-0.09%
deaths	0.1882	-81.18%
deaths^2	1.04	4.00%
deaths^3	0.9997	-0.03%
tower.dmg.1k	1.731	73.10%
tower.dmg.1k	0.9913	-0.87%
runes	0.9464	-5.36%

Table 5: Final Model Coefficients

How well does it predict a win though??
The *predict.glm()* function in R allows us to add a prediction column to the test data and see how well our model does on new data. In Figure 26 below, we find a confusion matrix that shows the various model statistics. With an accuracy of 0.969 (percentage of correct predictions), a sensitivity of 0.973 (also called precision or true positive rate), and a specificity of 0.965 (also called true negative rate), it seems like this model does quite well at predicting a match win.

- Sensitivity (how often were positive predictions correct) = TP / (TP + FP)
  = 674 / (674 + 19) = **0.973**
- Specificity (how often were negative predictions correct) = TN / (TN + FN)
  = 657 / (657 + 24) = **0.965**



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 657  19
         1  24 674

               Accuracy : 0.969
                 95% CI : (0.958, 0.977)
    No Information Rate : 0.504
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.937

 Mcnemar's Test P-Value : 0.542

            Sensitivity : 0.973
            Specificity : 0.965
         Pos Pred Value : 0.966
         Neg Pred Value : 0.972
             Prevalence : 0.504
         Detection Rate : 0.491
   Detection Prevalence : 0.508
      Balanced Accuracy : 0.969

       'Positive' Class : 1
```

Table 6: Model Predictions Confusion Matrix

Value obtained from this study

While one can argue about whether video games or eSports are important in today's world, there is no doubt that there are many millions of people that enjoy video games and some play competitively. I collected about 1,000 professional Dota 2 matches and found many insights that highlight important components of a winning strategy – like working with your team to kill your enemies (*assists*), avoiding *deaths*, and hitting their towers a lot (*tower damage*). In performing this analysis, I did a fair amount of research about creating and refining models and interpreting their results while practicing EDA and good visual design. This project has added value to my own life through improved knowledge and skills in many parts of the data science process. Due to my limited hardware and software resources at home, I could learn about dealing with big data, but could not physically practice working with big data through NoSQL or its alternatives.

Explanation of Terms

- Abilities – special actions a hero can take with varying cooldown
- Assists – damaged an enemy hero right before they were killed, but did not make the final blow to kill the enemy hero
- Deaths – an enemy hero or small minion damages a hero more than their total health, causes hero to lose gold and sit out a temporary amount of time (5 – 120 seconds)
- Dire – the top right of the battle arena where trees are bare and flowers dead
- Dota 2 – a popular eSports multiplayer online battle arena video game from Valve
- Experience (XP) – killing enemies or small minions gains a hero experience making the hero and its abilities stronger over time
- Gold – in match currency earned by heroes for killing enemies or small minions, can be traded for items or power ups that enhance hero's abilities
- First blood – the first kill in a Dota 2 match
- Hero – a character controlled by each player that grows stronger throughout a match by killing smaller minions and enemy heroes to accumulate gold and experience
- Kills – damaging an enemy hero more than their total health, earning the killer and his teammate accomplices gold and experience
- Kill/Death/Assist ratio (KDA) – a rough measure of a player's performance or match contribution calculated by adding kills and assists together and then dividing by deaths
- Observer wards – little lamps that illuminate parts of the arena for a short period of time giving one team an advantage in sight and information on where the enemies may be hiding
- Radiant – the bottom left side of the battle arena where trees grow tall and flowers blossom
- Runes – power ups that spawn near the middle of the map periodically to give heroes a small temporary boost in some trait – making a hero run very fast, going invisible, doubling a hero's attack damage, regenerating health, etc
- Towers – physical structures protecting parts of the arena, damaging them will eventually destroy them and lead to victory

Code used in this analysis

Python was used for data collection, cleaning, and preparation, while R was used for analyses and visualizations. All code can be viewed on my github at https://github.com/doug-cady/gmu_daen/tree/master/AIT580/Analysis_Project

References:

FAQ: How do I interpret odds ratios in logistic regression?. UCLA Statistical Consulting Group. Retrieved 5 March 2021, from https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-interpret-odds-ratios-in-logistic-regression/

Independent Samples T-test in R | Statistical Methods. Retrieved 5 March 2021, from https://stat-methods.com/home/independent-samples-t-test-r-2/

Koh, C. (2020). Interpreting results from logistic regression in R using Titanic dataset. Retrieved 5 March 2021, from https://medium.com/@conankoh/interpreting-results-from-logistic-regression-in-r-using-titanic-dataset-bb9f9a1f644c

Zhang, Z. (2016). Residuals and regression diagnostics: focusing on logistic regression. *Annals Of Translational Medicine*, *4*(10), 195-195. doi: 10.21037/atm.2016.03.36