Midterm Exam

Yifan Liu

11/5/2020

Instruction

This is your midterm exam that you are expected to work on it alone. You may NOT discuss any of the content of your exam with anyone except your instructor. This includes text, chat, email and other online forums. We expect you to respect and follow the GRS Academic and Professional Conduct Code.

Although you may NOT ask anyone directly, you are allowed to use external resources such as R codes on the Internet. If you do use someone's code, please make sure you clearly cite the origin of the code.

When you finish, please compile and submit the PDF file and the link to the GitHub repository that contains the entire analysis.

Introduction

In this exam, you will act as both the client and the consultant for the data that you collected in the data collection exercise (20pts). Please note that you are not allowed to change the data. The goal of this exam is to demonstrate your ability to perform the statistical analysis that you learned in this class so far. It is important to note that significance of the analysis is not the main goal of this exam but the focus is on the appropriateness of your approaches.

Data Description (10pts)

Please explain what your data is about and what the comparison of interest is. In the process, please make sure to demonstrate that you can load your data properly into R.

This analysis wants to explore the relationship between the appearance of certain class of champions in ARAM mode of the online video game League of Legends and the win rate of each game . The whole dataset is collected from each game the provider had played by himself or with his friends. Since there are two types of games (solo and pre-made) exist, this project wants to compare the win rate of solo games and pre-made games including the considerations of which class of champions contribute the most to the win rate. The match history recorded on OP.GG can serve as a proof of originality. Each row in the dataset represents one game and all the five champions on the data provider's side are included (exclude the opponent team). Below is the variable explanation:

- `solo`: binary data; whether this game was played by the provider alone or by he and his friends
- `assassin`: numeric data, the number of champions' appearance; the class of champions with high mobility who specialize in single target burst damage
- `fighter`: numeric data, the number of champions' appearance; the class of champions with short-ranged combatants who excel at both dealing and surviving damage
- `mage`: numeric data, the number of champions' appearance; the class of champions who possess great reach, ability-based area of effect damage and crowd control
- `marksman`: numeric data, the number of champions' appearance; the class of champions whose power almost exclusively revolves around their basic attacks
- `support`: numeric data, the number of champions' appearance; the class of champions who oversee the battlefield by protecting and opening up opportunities for their allies

- `tank`: numeric data, the number of champions' appearance; the class of champions who excel in shrugging off incoming damage and focus on disrupting their enemies more than being significant damage threats themselves
- `win`: binary data; whether this game was won or lost

```
league = read.csv("League_champ.csv")
names(league)[1] = "solo"
head(league)
```

```
##   solo assassin fighter mage marksman support tank win
## 1    1        0       1    2        0       2    0   0
## 2    1        0       0    1        4       0    0   0
## 3    1        0       1    2        2       0    0   1
## 4    1        1       2    1        1       0    0   1
## 5    1        0       1    1        3       0    0   0
## 6    1        0       2    0        2       1    0   1
```

EDA (10pts)

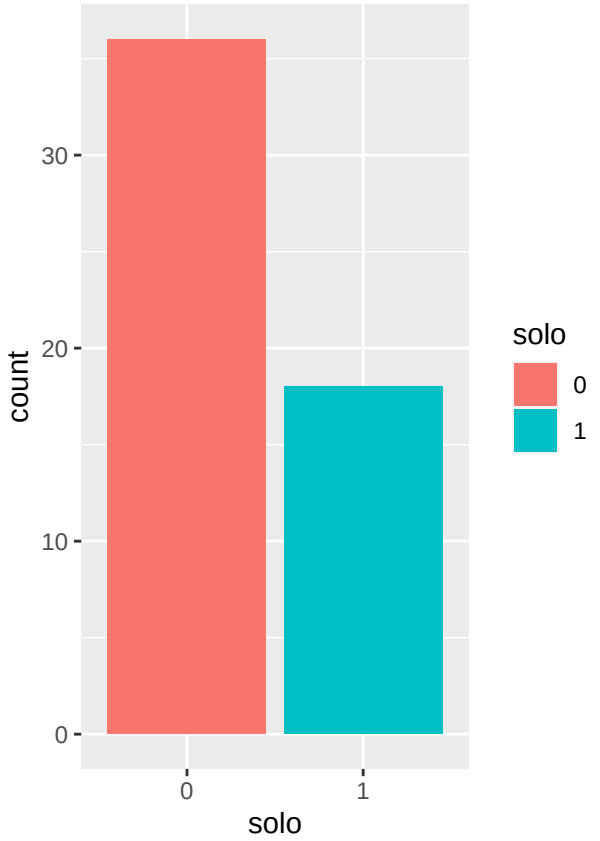
Please create one (maybe two) figure(s) that highlights the contrast of interest. Make sure you think ahead and match your figure with the analysis. For example, if your model requires you to take a log, make sure you take log in the figure as well.

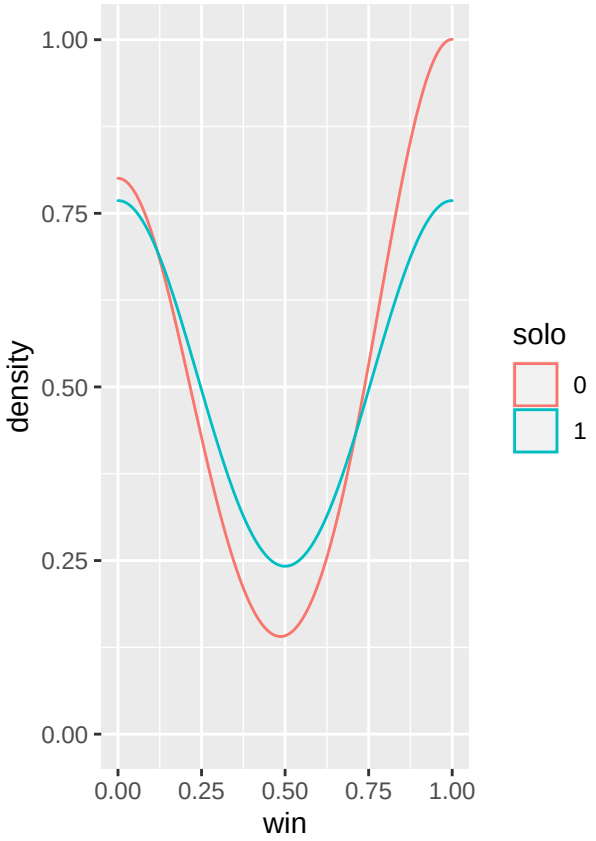
```
league$solo = as.factor(league$solo)
league_solo = filter(league, solo==1)
league_prem = filter(league, solo==0)
```

```
p01 = league %>%
  ggplot(aes(x = solo, fill = solo)) +
  geom_bar()

p02 = league %>%
  ggplot(aes(x = win, col = solo)) +
  geom_density()
```

```
grid.arrange(p01, p02, ncol = 2)
```





Power Analysis (10pts)

Please perform power analysis on the project. Use 80% power, the sample size you used and infer the level of effect size you will be able to detect. Discuss whether your sample size was enough for the problem at hand. Please note that method of power analysis should match the analysis. Also, please clearly state why you should NOT use the effect size from the fitted model.

```
pwr.f2.test(u = 6, v = 47, f2 = NULL, sig.level = 0.05, power = 0.8)
```

```
##
##      Multiple regression power calculation
##
##               u = 6
##               v = 47
##              f2 = 0.2876622
##       sig.level = 0.05
##           power = 0.8
```

After performing power analysis on the project using 80% power and the actual sample size, the inferred level of effect size that will be able to detect is around 0.29. In other words, this experiment will only have around 29% magnitude of effect. Generally speaking, the effect size will be expected to achieve around 0.5. According to this, the sample size in this project is clearly not enough which means more observations are needed in order to make a better analysis. The reason that the effect size from the fitted model should not be used is that due to the sample size, the effect size is really small and the power used to infer the effect size is also hypothetical which might not be true in the real analysis.

Modeling (10pts)

Please pick a regression model that best fits your data and fit your model. Please make sure you describe why you decide to choose the model. Also, if you are using GLM, make sure you explain your choice of link function as well.

The model that has been chosen is the logistic regression model. Because the predictors in the dataset are most numeric data with one categorical data. The predicted output can be viewed as the winning rate which equals to the number of winning divided by the total number of games. Also, the interest of this project is to explore the probability of winning. Thus the link function in the GLM will be used as "logit".

```
set.seed(1000)
fit = stan_glm(win ~ factor(solo) + assassin + fighter + mage + marksman + support + tank, family=binom
summary(fit)
```

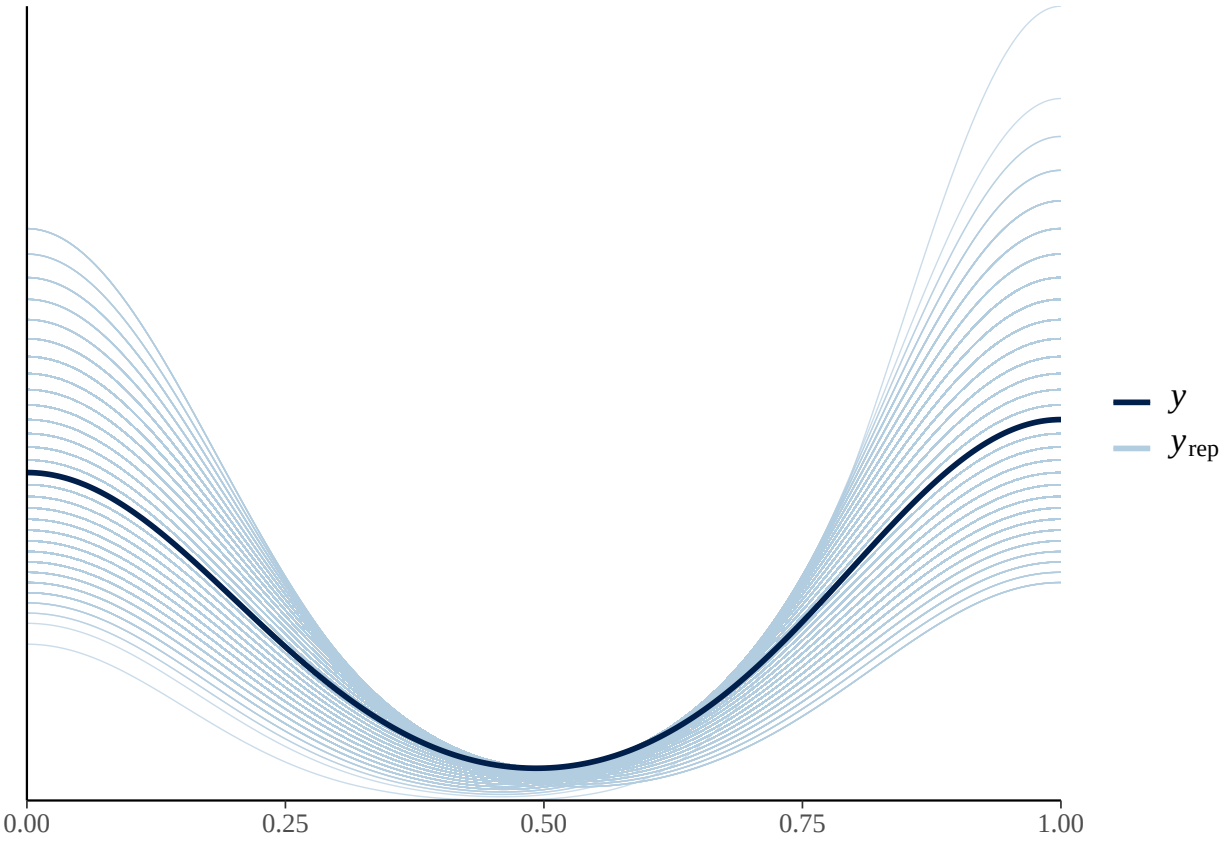
```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      win ~ factor(solo) + assassin + fighter + mage + marksman + support +
##     tank
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 54
##  predictors:   8
##
## Estimates:
##                  mean   sd   10%   50%   90%
## (Intercept)     1.3    7.5 -8.3   1.4  10.8
## factor(solo)1 -0.4    0.7 -1.3  -0.4   0.5
## assassin      -1.4    1.7 -3.5  -1.4   0.8
## fighter        0.5    1.5 -1.4   0.5   2.5
## mage           0.0    1.5 -2.0   0.0   2.0
## marksman      -0.4    1.5 -2.3  -0.4   1.6
## support       -1.4    1.6 -3.4  -1.4   0.6
## tank           7.6    4.2  2.7   7.0  13.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.1  0.4   0.5   0.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.3  1.0   727
## factor(solo)1 0.0  1.0  2287
## assassin      0.1  1.0   870
## fighter       0.1  1.0   798
## mage          0.1  1.0   769
## marksman      0.1  1.0   729
## support       0.1  1.0   753
## tank          0.1  1.0  1151
## mean_PPD      0.0  1.0  4010
## log-posterior 0.1  1.0  1150
```

```
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

Validation (10pts)

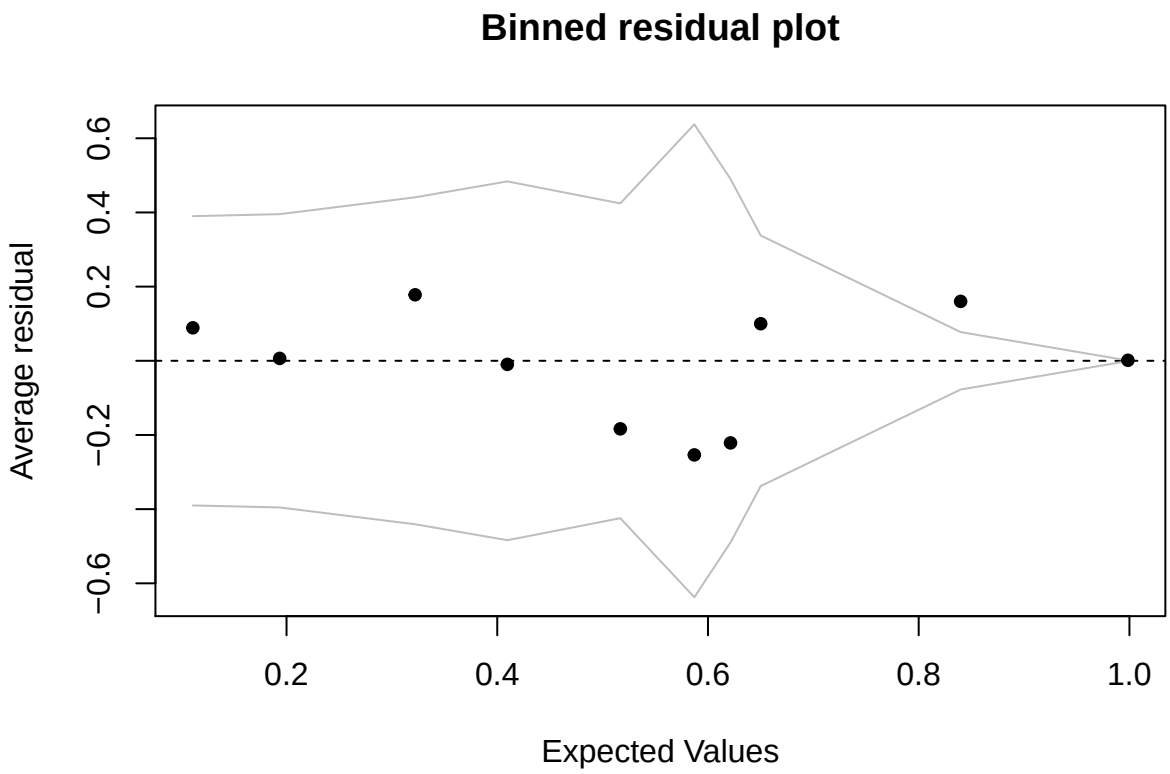
Please perform a necessary validation and argue why your choice of the model is appropriate.

```
pred = posterior_predict(fit)
ppc_dens_overlay(y=league$win, yrep = pred)
```



From the plot we can see that the model seems to predict the data well. Since the actual observations of `win` variable contains only 0 and 1, and the predicted values in 0 and 1 generally cover the actual values.

```
binnedplot(fitted(fit), resid(fit))
```



After checking the binned residual plot, the average residuals are generally close to zero and almost all residuals fall inside the error bound. This implies the original model is a suitable model.

Inference (10pts)

Based on the result so far please perform statistical inference to compare the comparison of interest.

```
fit_s = glm(win ~ assassin + fighter + mage + marksman + support + tank, data = league_solo)
summary(fit_s)
```

```
##
## Call:
## glm(formula = win ~ assassin + fighter + mage + marksman + support +
##     tank, data = league_solo)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.63412  -0.30710   0.00426   0.24688   0.64724
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.6006     2.6152   2.142   0.0534 .
## assassin     -1.2238     0.6777  -1.806   0.0961 .
## fighter      -0.7579     0.5007  -1.513   0.1560
## mage         -1.0772     0.5505  -1.957   0.0740 .
## marksman     -1.1498     0.5467  -2.103   0.0572 .
## support      -1.1858     0.5274  -2.248   0.0441 *
## tank             NA         NA      NA       NA
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.186269)
##
##     Null deviance: 4.5000  on 17  degrees of freedom
## Residual deviance: 2.2352  on 12  degrees of freedom
## AIC: 27.533
##
## Number of Fisher Scoring iterations: 2
```

In the solo matched games, we can see that the assassin class of champions have the most profound impact on the winning conditions in each game. In contrast, the fighter class of champions have the least impact on the winning conditions in each game. Because the appearance of tank class champions is so rare that the effect is neglected by the model. Suppose there is a solo matched game where each type of champions all appear except for tank, then the winning rate will be 0.2061.

```
fit_p = glm(win ~ assassin + fighter + mage + marksman + support + tank, data = league_prem)
summary(fit_p)
```

```
##
## Call:
## glm(formula = win ~ assassin + fighter + mage + marksman + support +
##     tank, data = league_prem)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.59594  -0.43917  -0.05249   0.29969   0.80521
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.3201     1.1097   2.992  0.00550 **
## assassin     -0.8062     0.3096  -2.604  0.01420 *
## fighter      -0.5414     0.2412  -2.245  0.03231 *
## mage         -0.4749     0.2361  -2.012  0.05330 .
## marksman     -0.5265     0.2340  -2.250  0.03192 *
## support      -0.7580     0.2507  -3.024  0.00508 **
## tank             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2159209)
##
##     Null deviance: 8.8889  on 35  degrees of freedom
## Residual deviance: 6.4776  on 30  degrees of freedom
## AIC: 54.418
##
## Number of Fisher Scoring iterations: 2
```

In the pre-made matched games, we can see that the assassin class of champions still have the most profound impact on the winning conditions in each game. However in this type of games, the mage class of champions have the least impact on the winning conditions in each game. And compared to solo matched games, the difference between the impacts brought by different class of champions is relatively small in pre-made matched games. Also, because the appearance of tank class champions is so rare that the effect is neglected by the model. Suppose there is a pre-made matched game where each type of champions all appear except for tank, then the winning rate will be 0.2131.

Discussion (10pts)

Please clearly state your conclusion and the implication of the result.

Given the results above, the logistic regression model seems to fit well to this dataset. According to the outputs, in general, when the formation of class of champions is the same, the pre-made matched games will have a slightly higher probability of winning than the solo matched games. To be more specific, in solo matched games, the difference brought by the different class of champions is relatively large which means the component of the class of champions might have a greater impact on the winning conditions of that game. In contrast, the difference brought by the different class of champions in pre-made matched games is relative small which means the component of the class of champions might not be as important as in solo matched games while other factors such as team cooperation might come into play. It appears that in both situations, assassin class of champions have the most profound impacts on the winning conditions while fighter and mage class of champions generally have the least impacts on the winning conditions. This might brought to the further exploration that whether it is the skills of assassin players that contributes to this influence or it is the mechanism of this type of champions that does so. Also, the reason for the little appearance of tank champions might need further investigations.

Limitations and future opportunity. (10pts)

Please list concerns about your analysis. Also, please state how you might go about fixing the problem in your future study.

The major concern for this analysis is the data size. As mentioned before, the dataset is collected by the provider manually and it took a long time to perform all the 54 games even though this size is extremely small for data analysis. One evidence is that since the tank champions appear too less times, the model treat the `tank` variable as NA values and the outputs become all NA values. Also, there are other uncertainties that mind influence the data such as the players' skills and attitudes while they were playing the games. These uncertainties need further considerations for a better analysis. In order to fix the problem, one way that can be performed is to acquire data and player information from the relative sources such as from the department of Technical Support of League of Legends or any other third-party websites. Also, recording the data in other forms might suit the dataset better to the model that has been created.

Comments or questions

If you have any comments or questions, please write them here.