Data Mining on Dota 2 Matches

Tianyu Lu

April 11, 2016

Abstract

Dota (Defense of the Ancient) 2 is a combination of RTS including perspective and a heavy requirement of tactics and team co-ordination and RPG including itemization and leveling up. This paper applies different model to generate analysis of the skill stats in the match and draws a series conclusions to help players and teams to establish their strategy. In the future, with the result in this paper, it is possible to get a model for match predicting. The model can calculate a static score for the teams. And then, apply to statistical model.

Background

Dota (Defense of the Ancient) 2 is a combination of RTS including perspective and a heavy requirement of tactics and team co-ordination and RPG including itemization and leveling up. Players are split into two competing teams (Radiant and Dire), each consisting of up to five players. The main objective in Dota 2 is to destroy the enemy Ancient inside their stronghold. These strongholds are protected by multiple towers down 3 lanes. The player controls a Hero, a strategically-powerful unit with unique abilities and characteristics, which fights for them and gains strength by leveling up and buying items with gold. Experience is earned when creeps and heroes die. Gold is gained passively over time, by killing creeps, by killing enemy heroes and by destroying buildings.

Heroes are all unique characters within Dota 2. At the start of each game, players select a Hero from the Hero Pool. Heroes are split into 3 categories: strength, agility and intelligence. Dota 2 has currently implemented 111 of the 112 Heroes.

It is a multiplayer online battle arena video game published by Valve Corporation. Every year Valve hosts a great electronic sports Dota 2 championship tournament: The International. It attracts many different players from all over the world, with a large prize pool (In 2015, it was $18 million). Also, millions of players played the game online every day. It is a team fight between two five-player teams, each of which occupies a stronghold in a corner of the map. Who destroy the Ancient on the other side wins the game. It has more than 100 heroes and complicated item system. So, for this big game, players need a data analysis report to guide them to play it well. All the teams need data mining process to see how to play against their opponent including which hero should be banned, which should be picked and how to make a strategy to form a hero group. Currently, teams has their data analyzer help to make a basic statistic on which hero has higher win rate and how to combine a strategy with hero picked in a game. They may also try to study their opponent's preference on hero selecting. But in whole Dota field, it lacks a deeper data mining with professional statistic methods applied in economy or other serious subject. Players always use their own feeling to pick hero in a match or just make a simple ban/pick strategy. So, in this project, we will use machine learning method to find out those important factors and make a prediction for the match. In this way, if the result is satisfactory, it can generate a model for teams to build up their strategy when facing different opponents.

Data Collection and Description

I collected match data from http://www.dotabuff.com/esports/matches. It contains the details of matches of the game and some important statistics such as damage, experience and gold of each hero. I picked the data

only from recent esports games, so it reflects the high level skill players' strategy, which can give us more confident machine learning result. These matches are under the newest version of Dota 2 6.86f.

The attribute information is shown as below:

- 1. Match ID - Describe the identification of the match.
- 2. Region - The match server location.
- 3. Duration - The time period of the match, excluding the ban/pick time.
- 4. Winner - Radiant or Dire.
- 5. Hero - The hero picked in the game.
- 6. K - For one hero, the number of enemy hero he kills in the game.
- 7. D - For one hero, the time he dies in the game.
- 8. A - For one hero, the time he assists to kill an enemy hero in the game.
- 9. XPM - The experience per minute of one hero.
- 10. GPM - The gold per minute of one hero.
- 11. DMG - The total damage of one hero deals in the game.

(From 5 to 11, these numbers describe one hero's performance in a game. Since it has 10 heroes, 5 vs 5, in the match, we use number 1 to 10 to express each group of statistics of one hero. And 1 to 5 means radiant hero, whereas 6 to 10 dire.)

Algorithm and Model

For the data, we are going to apply different algorithms to analyze what are the main factors that influence the match. Hence, we can draw conclusions of how to win the game and how to operate different kind heros.

0. Data Preprocessing

The raw data lacks some hero in the game, so we should do some work to uniform the hero levels, which can help when we apply the hero analysis.

```
dota.orgin <- read.csv(file = "D:/dotabuff.csv",header = T)
summary(dota.orgin)
```

```
##        ID                     Region       Duration       Winner
##  Min.   :2.274e+09   China        :318   15:09  :  3   Dire   :472
##  1st Qu.:2.277e+09   Europe West  :335   18:47  :  3   Radiant:494
##  Median :2.283e+09   Russia       :  6   23:09  :  3
##  Mean   :2.282e+09   SE Asia      : 92   23:27  :  3
##  3rd Qu.:2.286e+09   South America: 30   25:11  :  3
##  Max.   :2.291e+09   US East      :165   27:40  :  3
##                      US West      : 20   (Other):948
##              Hero1           K1              D1              A1
##  invoker        : 37   Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
##  death-prophet  : 36   1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 6.00
```

```
##  lion           : 35   Median : 4.000   Median : 5.000   Median :10.00
##  vengeful-spirit: 34   Mean   : 5.101   Mean   : 5.521   Mean   :11.09
##  witch-doctor   : 33   3rd Qu.: 7.000   3rd Qu.: 8.000   3rd Qu.:16.00
##  earth-spirit   : 26   Max.   :31.000   Max.   :23.000   Max.   :41.00
##  (Other)        :765
##       XPM1          GPM1            DMG1                Hero2
##  -      : 14   Min.   :109.0   4.9k   : 20   witch-doctor   : 41
##  332    :  9   1st Qu.:277.0   6.2k   : 18   invoker        : 39
##  396    :  9   Median :358.0   -      : 16   lion           : 37
##  194    :  7   Mean   :387.5   5.6k   : 15   vengeful-spirit: 35
##  285    :  7   3rd Qu.:494.2   4.2k   : 14   doom           : 29
##  246    :  6   Max.   :986.0   4.3k   : 12   spectre        : 27
##  (Other):914                   (Other):871   (Other)        :758
##        K2              D2              A2             XPM2
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.00   -      : 13
##  1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 5.00   369    :  9
##  Median : 4.000   Median : 5.000   Median :10.00   392    :  7
##  Mean   : 5.245   Mean   : 5.171   Mean   :11.16   400    :  7
##  3rd Qu.: 7.000   3rd Qu.: 7.000   3rd Qu.:16.00   401    :  7
##  Max.   :32.000   Max.   :22.000   Max.   :46.00   279    :  6
##                                                    (Other):917
##       GPM2          DMG2                Hero3           K3
##  328    :  8   -      : 15   invoker        : 47   Min.   : 0.000
##  368    :  7   4.7k   : 15   zeus           : 33   1st Qu.: 2.000
##  231    :  6   5.3k   : 15   witch-doctor   : 32   Median : 4.000
##  306    :  6   3.8k   : 14   spectre        : 30   Mean   : 5.306
##  325    :  6   7.9k   : 14   vengeful-spirit: 29   3rd Qu.: 8.000
##  377    :  6   4.1k   : 13   natures-prophet: 28   Max.   :26.000
##  (Other):927   (Other):880   (Other)        :767
##        D3              A3             XPM3          GPM3
##  Min.   : 0.000   Min.   : 0.00   -      : 13   301    :  8
##  1st Qu.: 3.000   1st Qu.: 5.25   336    : 10   276    :  7
##  Median : 5.000   Median :10.00   513    :  8   464    :  7
##  Mean   : 5.306   Mean   :11.06   434    :  7   231    :  6
##  3rd Qu.: 7.000   3rd Qu.:15.00   249    :  6   275    :  6
##  Max.   :21.000   Max.   :39.00   296    :  6   395    :  6
##                                   (Other):916   (Other):926
##       DMG3                Hero4           K4              D4
##  -      : 14   witch-doctor   : 38   Min.   : 0.00   Min.   : 0.000
##  5.9k   : 13   invoker        : 35   1st Qu.: 2.00   1st Qu.: 3.000
##  6.2k   : 13   vengeful-spirit: 33   Median : 4.00   Median : 5.000
##  6.4k   : 13   lion           : 31   Mean   : 5.18   Mean   : 5.287
##  4.3k   : 12   spectre        : 31   3rd Qu.: 8.00   3rd Qu.: 8.000
##  4.4k   : 12   sven           : 29   Max.   :30.00   Max.   :18.000
##  (Other):889   (Other)        :769
##        A4             XPM4          GPM4          DMG4
##  Min.   : 0.00   -      : 13   266    :  8   -      : 15
##  1st Qu.: 5.00   315    :  7   347    :  8   4.1k   : 15
##  Median :10.00   178    :  6   245    :  7   5k     : 15
##  Mean   :10.76   270    :  6   256    :  7   4.4k   : 14
##  3rd Qu.:15.00   272    :  6   326    :  7   3.7k   : 13
##  Max.   :45.00   297    :  6   231    :  6   4.5k   : 13
##                  (Other):922   (Other):923   (Other):881
##              Hero5           K5              D5              A5
```

```
##  invoker        : 37   Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
##  beastmaster    : 33   1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 5.00
##  vengeful-spirit: 32   Median : 4.000   Median : 5.000   Median :10.00
##  lion           : 29   Mean   : 4.873   Mean   : 5.212   Mean   :10.91
##  witch-doctor   : 29   3rd Qu.: 7.000   3rd Qu.: 7.000   3rd Qu.:15.00
##  faceless-void  : 26   Max.   :32.000   Max.   :17.000   Max.   :54.00
##  (Other)        :780
##       XPM5          GPM5            DMG5                Hero6
##  -      : 12   Min.   :116.0   4.5k   : 16   invoker        : 38
##  338    :  7   1st Qu.:279.0   -      : 14   lion           : 36
##  201    :  6   Median :367.0   4.3k   : 14   witch-doctor   : 33
##  276    :  6   Mean   :387.7   3.2k   : 13   vengeful-spirit: 32
##  320    :  6   3rd Qu.:489.0   3.9k   : 13   death-prophet  : 27
##  376    :  6   Max.   :877.0   9.3k   : 13   faceless-void  : 26
##  (Other):923                   (Other):883   (Other)        :774
##        K6              D6              A6             XPM6
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.00   -      : 12
##  1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 5.00   386    :  7
##  Median : 4.000   Median : 5.000   Median :10.00   303    :  6
##  Mean   : 5.081   Mean   : 5.407   Mean   :11.24   334    :  6
##  3rd Qu.: 7.000   3rd Qu.: 7.750   3rd Qu.:16.00   359    :  6
##  Max.   :34.000   Max.   :21.000   Max.   :48.00   513    :  6
##                                                    (Other):923
##       GPM6          DMG6                  Hero7           K7
##  279    :  7   6.5k   : 20   invoker          : 43   Min.   : 0.000
##  239    :  6   -      : 15   outworld-devourer: 36   1st Qu.: 2.000
##  296    :  6   6.2k   : 15   witch-doctor     : 35   Median : 4.000
##  323    :  6   4.6k   : 13   juggernaut       : 31   Mean   : 5.827
##  338    :  6   5.4k   : 13   lion             : 31   3rd Qu.: 8.000
##  373    :  6   5k     : 13   beastmaster      : 30   Max.   :34.000
##  (Other):929   (Other):877   (Other)          :760
##        D7              A7             XPM7          GPM7
##  Min.   : 0.000   Min.   : 0.00   -      : 13   Min.   :105.0
##  1st Qu.: 3.000   1st Qu.: 5.00   359    :  7   1st Qu.:289.0
##  Median : 5.000   Median :10.00   345    :  6   Median :392.5
##  Mean   : 5.163   Mean   :10.73   397    :  6   Mean   :412.1
##  3rd Qu.: 7.000   3rd Qu.:15.00   489    :  6   3rd Qu.:528.0
##  Max.   :18.000   Max.   :38.00   564    :  6   Max.   :960.0
##                                   (Other):922
##       DMG7                 Hero8           K8              D8
##  -      : 15   lion           : 46   Min.   : 0.000   Min.   : 0.000
##  4.5k   : 13   invoker        : 42   1st Qu.: 2.000   1st Qu.: 3.000
##  5k     : 13   zeus           : 41   Median : 4.000   Median : 5.000
##  9.4k   : 12   doom           : 33   Mean   : 5.112   Mean   : 5.353
##  2.9k   : 11   natures-prophet: 27   3rd Qu.: 7.000   3rd Qu.: 7.000
##  3.8k   : 11   beastmaster    : 26   Max.   :27.000   Max.   :17.000
##  (Other):891   (Other)        :751
##        A8          XPM8          GPM8            DMG8
##  Min.   : 0   -      : 11   Min.   :105.0   -      : 15
##  1st Qu.: 5   317    :  7   1st Qu.:285.0   8.7k   : 15
##  Median :10   376    :  7   Median :369.0   4.8k   : 14
##  Mean   :11   379    :  7   Mean   :392.4   4.5k   : 13
##  3rd Qu.:15   340    :  6   3rd Qu.:491.0   5.8k   : 13
##  Max.   :47   344    :  6   Max.   :941.0   9.3k   : 13
```

```
##               (Other):922                    (Other):883
##              Hero9           K9               D9               A9
##  lion            : 51   Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
##  witch-doctor    : 40   1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 5.00
##  faceless-void   : 37   Median : 4.000   Median : 5.000   Median :10.00
##  invoker         : 33   Mean   : 4.757   Mean   : 5.232   Mean   :11.02
##  vengeful-spirit: 26   3rd Qu.: 7.000   3rd Qu.: 7.000   3rd Qu.:16.00
##  doom            : 25   Max.   :28.000   Max.   :21.000   Max.   :40.00
##  (Other)         :754
##       XPM9           GPM9             DMG9                  Hero10
##  -      : 12   Min.   :100.0   3.8k   : 16   witch-doctor   : 38
##  288    :  7   1st Qu.:272.0   4.1k   : 16   beastmaster    : 36
##  240    :  6   Median :355.0   7.9k   : 15   faceless-void  : 34
##  276    :  6   Mean   :381.0   -      : 14   lion           : 34
##  284    :  6   3rd Qu.:468.2   4.2k   : 14   bounty-hunter  : 28
##  290    :  6   Max.   :955.0   4.8k   : 13   vengeful-spirit: 27
##  (Other):923                    (Other):878   (Other)        :769
##       K10              D10              A10            XPM10
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.00   -      : 12
##  1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 5.00   398    :  7
##  Median : 4.000   Median : 5.000   Median :10.00   284    :  6
##  Mean   : 4.925   Mean   : 5.299   Mean   :11.26   294    :  6
##  3rd Qu.: 7.000   3rd Qu.: 7.000   3rd Qu.:16.00   359    :  6
##  Max.   :26.000   Max.   :19.000   Max.   :49.00   360    :  6
##                                                    (Other):923
##      GPM10         DMG10
##  375    :  9   -      : 15
##  252    :  7   4.5k   : 15
##  331    :  7   4.1k   : 14
##  282    :  6   4k     : 14
##  320    :  6   6.1k   : 14
##  357    :  6   6.6k   : 14
##  (Other):925   (Other):880
```

```
lvl <- c("abaddon", "alchemist", "ancient-apparition", "anti-mage", "arc-warden", "axe" ,"bane",  "batr

dota.orgin$Hero1 <- factor(as.character(dota.orgin$Hero1),levels = lvl)
dota.orgin$Hero2 <- factor(as.character(dota.orgin$Hero2),levels = lvl)
dota.orgin$Hero3 <- factor(as.character(dota.orgin$Hero3),levels = lvl)
dota.orgin$Hero4 <- factor(as.character(dota.orgin$Hero4),levels = lvl)
dota.orgin$Hero5 <- factor(as.character(dota.orgin$Hero5),levels = lvl)
dota.orgin$Hero6 <- factor(as.character(dota.orgin$Hero6),levels = lvl)
dota.orgin$Hero7 <- factor(as.character(dota.orgin$Hero7),levels = lvl)
dota.orgin$Hero8 <- factor(as.character(dota.orgin$Hero8),levels = lvl)
dota.orgin$Hero9 <- factor(as.character(dota.orgin$Hero9),levels = lvl)
dota.orgin$Hero10 <- factor(as.character(dota.orgin$Hero10),levels = lvl)
```

1. Logistic Resgression

From basic observation, I want to try linear regression to find out which parameters affect the match result mostly. It is a simple model to observe the inner relationship of duration, KDA, experience and gold with the win rate. Basically, we think the game has rationship with all the factors, especially the kda and the gold they earn from enemy heroes and minions.

Data convertion

First, we convert the data to numeric.

```
#convert duration to numeric
dota.orgin$Duration <- as.character(dota.orgin$Duration)
shortformat <- "^[0-9]{2}:[0-9]{2}$"
mark <- grep(shortformat,dota.orgin$Duration)
for(i in mark){
  dota.orgin$Duration[i] <- paste("00:",dota.orgin$Duration[i])
}
library(chron)
```

```
## Warning: package 'chron' was built under R version 3.2.4
```

```
duration <- chron(times = as.character(dota.orgin$Duration))
class(duration)
```

```
## [1] "times"
```

```
duration <- data.frame(matrix(unlist(strsplit(as.character(dota.orgin$Duration), split = ":")),ncol = 3
colnames(duration) <- c("hour","minute","second")
minutes <- as.numeric(as.character(duration$hour))*60+as.numeric(as.character(duration$minute))

#convert skill statistics to numeric format
dota.parameters <- dota.orgin[,c(-1:-5,-11,-12,-18,-19,-25,-26,-32,-33,-39,-40,-46,-47,-53,-54,-60,-61,
for(i in 1:50){
dota.parameters[,i] <- as.numeric(as.character(dota.parameters[,i]))
}
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```
dota.parameters <- cbind(Winner=dota.orgin$Winner,duration=minutes,dota.parameters)
dota.parameters <- na.omit(dota.parameters)
```

Logistic regression for all factors

It is easily to run logistic regression model with all the parameters. But since the sum of statistics for Hero1 to Hero5 have collinearity with Hero6 to Hero10, we separate them into 2 groups to generate 2 models.

```
dota.lm.radiant <- lm(as.numeric(Winner)~duration+
                                K1+D1+A1+XPM1+GPM1+
                                K2+D2+A2+XPM2+GPM2+
                                K3+D3+A3+XPM3+GPM3+
                                K4+D4+A4+XPM4+GPM4+
                                K5+D5+A5+XPM5+GPM5
                                #K6+D6+A6+XPM6+GPM6+
                                #K7+D7+A7+XPM7+GPM7+
                                #K8+D8+A8+XPM8+GPM8+
                                #K9+D9+A9+XPM9+GPM9+
                                #K10+D10+A10+XPM10+GPM10
                                , data = dota.parameters)
summary(dota.lm.radiant)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ duration + K1 + D1 + A1 + XPM1 +
##     GPM1 + K2 + D2 + A2 + XPM2 + GPM2 + K3 + D3 + A3 + XPM3 +
##     GPM3 + K4 + D4 + A4 + XPM4 + GPM4 + K5 + D5 + A5 + XPM5 +
##     GPM5, data = dota.parameters)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.92157 -0.17712  0.00345  0.17395  0.73901
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.104e-01  6.733e-02  12.036  < 2e-16 ***
## duration    -1.924e-03  1.051e-03  -1.831 0.067393 .
## K1           4.486e-03  3.306e-03   1.357 0.175181
## D1          -1.052e-02  4.059e-03  -2.591 0.009718 **
## A1           2.339e-03  2.304e-03   1.015 0.310340
## XPM1         2.133e-04  1.363e-04   1.565 0.117928
## GPM1         2.846e-04  1.433e-04   1.985 0.047402 *
## K2           1.015e-02  3.229e-03   3.144 0.001720 **
## D2          -2.387e-02  4.148e-03  -5.755 1.18e-08 ***
```

```
## A2           4.303e-03  2.379e-03   1.808 0.070862 .
## XPM2        -2.147e-04  1.477e-04  -1.454 0.146404
## GPM2         4.859e-04  1.579e-04   3.077 0.002153 **
## K3           8.288e-03  3.302e-03   2.510 0.012257 *
## D3          -1.240e-02  3.879e-03  -3.196 0.001441 **
## A3           7.345e-03  2.369e-03   3.101 0.001989 **
## XPM3         6.890e-05  1.541e-04   0.447 0.654836
## GPM3         4.341e-04  1.609e-04   2.699 0.007090 **
## K4          -4.578e-04  3.498e-03  -0.131 0.895920
## D4          -1.995e-02  4.004e-03  -4.982 7.53e-07 ***
## A4           4.977e-04  2.313e-03   0.215 0.829675
## XPM4         1.192e-05  1.409e-04   0.085 0.932606
## GPM4         5.413e-04  1.469e-04   3.686 0.000241 ***
## K5           8.213e-04  3.296e-03   0.249 0.803269
## D5          -1.716e-02  4.169e-03  -4.116 4.20e-05 ***
## A5           2.350e-03  2.220e-03   1.059 0.290048
## XPM5         6.924e-05  1.443e-04   0.480 0.631465
## GPM5         3.821e-04  1.558e-04   2.453 0.014364 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2583 on 913 degrees of freedom
## Multiple R-squared:  0.7407, Adjusted R-squared:  0.7333
## F-statistic: 100.3 on 26 and 913 DF,  p-value: < 2.2e-16
```

```
dota.lm.dire <- lm(as.numeric(Winner)~duration+
                                #K1+D1+A1+XPM1+GPM1+
                                #K2+D2+A2+XPM2+GPM2+
                                #K3+D3+A3+XPM3+GPM3+
                                #K4+D4+A4+XPM4+GPM4+
                                #K5+D5+A5+XPM5+GPM5+
                                K6+D6+A6+XPM6+GPM6+
                                K7+D7+A7+XPM7+GPM7+
                                K8+D8+A8+XPM8+GPM8+
                                K9+D9+A9+XPM9+GPM9+
                                K10+D10+A10+XPM10+GPM10
                                , data = dota.parameters)
summary(dota.lm.dire)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ duration + K6 + D6 + A6 + XPM6 +
##     GPM6 + K7 + D7 + A7 + XPM7 + GPM7 + K8 + D8 + A8 + XPM8 +
##     GPM8 + K9 + D9 + A9 + XPM9 + GPM9 + K10 + D10 + A10 + XPM10 +
##     GPM10, data = dota.parameters)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.07088 -0.16209 -0.00186  0.17025  0.72162
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.335e+00  6.014e-02  38.823  < 2e-16 ***
## duration     2.016e-03  1.037e-03   1.945 0.052128 .
```

```
## K6          -8.686e-03  3.198e-03  -2.716 0.006729 **
## D6           1.946e-02  3.976e-03   4.894 1.17e-06 ***
## A6          -7.185e-03  2.252e-03  -3.191 0.001465 **
## XPM6        -4.163e-05  1.369e-04  -0.304 0.761165
## GPM6        -4.198e-04  1.425e-04  -2.945 0.003311 **
## K7          -5.169e-03  2.966e-03  -1.743 0.081743 .
## D7           1.395e-02  3.915e-03   3.563 0.000386 ***
## A7          -3.774e-03  2.372e-03  -1.591 0.112012
## XPM7        -1.542e-04  1.316e-04  -1.172 0.241573
## GPM7        -3.038e-04  1.385e-04  -2.193 0.028536 *
## K8          -2.230e-04  3.213e-03  -0.069 0.944691
## D8           1.838e-02  3.945e-03   4.658 3.67e-06 ***
## A8          -1.043e-03  2.204e-03  -0.473 0.636072
## XPM8        -8.555e-05  1.473e-04  -0.581 0.561577
## GPM8        -3.966e-04  1.479e-04  -2.682 0.007448 **
## K9           1.408e-03  3.423e-03   0.411 0.680913
## D9           9.473e-03  4.162e-03   2.276 0.023076 *
## A9          -4.294e-03  2.169e-03  -1.980 0.047986 *
## XPM9        -8.472e-05  1.442e-04  -0.588 0.556905
## GPM9        -4.334e-04  1.471e-04  -2.946 0.003296 **
## K10          1.953e-03  3.312e-03   0.590 0.555634
## D10          1.416e-02  4.106e-03   3.450 0.000587 ***
## A10         -5.203e-03  2.286e-03  -2.276 0.023069 *
## XPM10        1.378e-04  1.514e-04   0.910 0.362841
## GPM10       -7.479e-04  1.527e-04  -4.898 1.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2535 on 913 degrees of freedom
## Multiple R-squared:  0.7502, Adjusted R-squared:  0.7431
## F-statistic: 105.5 on 26 and 913 DF,  p-value: < 2.2e-16
```

```
cbind(dota.lm.radiant$coefficients,dota.lm.dire$coefficients)
```

```
##                      [,1]          [,2]
## (Intercept)  8.103815e-01  2.334638e+00
## duration    -1.923817e-03  2.016363e-03
## K1           4.486091e-03 -8.685770e-03
## D1          -1.051634e-02  1.945700e-02
## A1           2.338824e-03 -7.185042e-03
## XPM1         2.133061e-04 -4.163440e-05
## GPM1         2.845958e-04 -4.198036e-04
## K2           1.015251e-02 -5.168833e-03
## D2          -2.387342e-02  1.394935e-02
## A2           4.302589e-03 -3.773848e-03
## XPM2        -2.147067e-04 -1.542373e-04
## GPM2         4.859343e-04 -3.038542e-04
## K3           8.287641e-03 -2.229677e-04
## D3          -1.239904e-02  1.837801e-02
## A3           7.345335e-03 -1.043328e-03
## XPM3         6.889725e-05 -8.555047e-05
## GPM3         4.341229e-04 -3.966236e-04
## K4          -4.577703e-04  1.407874e-03
## D4          -1.994647e-02  9.473012e-03
```

```
## A4           4.976812e-04 -4.294130e-03
## XPM4         1.191759e-05 -8.472471e-05
## GPM4         5.412925e-04 -4.334022e-04
## K5           8.212783e-04  1.952820e-03
## D5          -1.716082e-02  1.416268e-02
## A5           2.350322e-03 -5.203336e-03
## XPM5         6.924266e-05  1.378226e-04
## GPM5         3.821186e-04 -7.479295e-04
```

From this two linear model, we can know the duration and XPM are not significant to the win rate, though the skill numbers get larger when the game last longer. It is not a good model for us to anlysis, but anyway, at least we know the GPM and Death number are important. After comparing the coefficients of both side, I think the distribution of the skill statistics is not balance, for one hero from a certain side may get all the killings and earn much more gold than his teammates.

To avoid this bias, we should add the numbers together to get the total number for each side.

Logistic regression for sums

```
#the sum for radiant heroes
Kradiant <- dota.parameters$K1+dota.parameters$K2+dota.parameters$K3+dota.parameters$K4+dota.parameters$
Dradiant <- dota.parameters$D1+dota.parameters$D2+dota.parameters$D3+dota.parameters$D4+dota.parameters$
Aradiant <- dota.parameters$A1+dota.parameters$A2+dota.parameters$A3+dota.parameters$A4+dota.parameters$
Xradiant <- dota.parameters$XPM1+dota.parameters$XPM2+dota.parameters$XPM3+dota.parameters$XPM4+dota.pa
Gradiant <- dota.parameters$GPM1+dota.parameters$GPM2+dota.parameters$GPM3+dota.parameters$GPM4+dota.pa
#the sum for dire heroes
Kdire <- dota.parameters$K6+dota.parameters$K7+dota.parameters$K8+dota.parameters$K9+dota.parameters$K1
Ddire <- dota.parameters$D6+dota.parameters$D7+dota.parameters$D8+dota.parameters$D9+dota.parameters$D1
Adire <- dota.parameters$A6+dota.parameters$A7+dota.parameters$A8+dota.parameters$A9+dota.parameters$A1
Xdire <- dota.parameters$XPM6+dota.parameters$XPM7+dota.parameters$XPM8+dota.parameters$XPM9+dota.param
Gdire <- dota.parameters$GPM6+dota.parameters$GPM7+dota.parameters$GPM8+dota.parameters$GPM9+dota.param

dota.sum <- cbind(dota.parameters[,1:2],Kradiant,Dradiant,Aradiant,Xradiant,Gradiant,Kdire,Ddire,Adire,X
head(dota.sum)
```

```
##    Winner duration Kradiant Dradiant Aradiant Xradiant Gradiant Kdire
## 1    Dire       23       12       27       24     1520     1619    26
## 2 Radiant       50       55       30      113     2496     2162    30
## 3 Radiant       43       27       16       70     2100     1981    15
## 4    Dire       28        8       34       15     1174     1160    34
## 5 Radiant       26       25        9       69     2115     2218     9
## 6 Radiant       45       54       46       65     2831     2477    45
##   Ddire Adire Xdire Gdire
## 1    13    79  2318  2389
## 2    56    62  2066  1771
## 3    27    36  1895  1903
## 4     8    94  1972  2105
## 5    25    24  1589  1590
## 6    54    96  2864  2366
```

Now we get the sum data frame for new linear regression models:

```
dota.lm.radiant.sum <- lm(as.numeric(Winner)~duration+Kradiant+Dradiant+Aradiant+Xradiant+Gradiant,dota
summary(dota.lm.radiant.sum)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ duration + Kradiant + Dradiant +
##     Aradiant + Xradiant + Gradiant, data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90877 -0.17316  0.00172  0.16895  0.73958
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.018e-01  6.721e-02  11.929  < 2e-16 ***
## duration    -1.691e-03  1.028e-03  -1.645  0.10026
## Kradiant     5.281e-03  1.999e-03   2.641  0.00839 **
## Dradiant    -1.671e-02  8.871e-04 -18.833  < 2e-16 ***
## Aradiant     3.225e-03  7.665e-04   4.208 2.83e-05 ***
## Xradiant     2.489e-05  4.452e-05   0.559  0.57624
## Gradiant     4.283e-04  4.395e-05   9.745  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.259 on 933 degrees of freedom
## Multiple R-squared:  0.7335, Adjusted R-squared:  0.7318
## F-statistic:   428 on 6 and 933 DF,  p-value: < 2.2e-16
```

```
dota.lm.dire.sum <- lm(as.numeric(Winner)~duration+Kdire+Ddire+Adire+Xdire+Gdire,dota.sum)
summary(dota.lm.dire.sum)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ duration + Kdire + Ddire +
##     Adire + Xdire + Gdire, data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25516 -0.17583 -0.00279  0.17169  0.70191
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.319e+00  5.977e-02  38.799  < 2e-16 ***
## duration     2.371e-03  1.020e-03   2.325   0.0203 *
## Kdire       -2.902e-03  1.868e-03  -1.554   0.1205
## Ddire        1.509e-02  8.792e-04  17.169  < 2e-16 ***
## Adire       -4.195e-03  6.880e-04  -6.098 1.57e-09 ***
## Xdire       -5.286e-05  4.471e-05  -1.182   0.2374
## Gdire       -4.452e-04  4.416e-05 -10.083  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.2538 on 933 degrees of freedom
## Multiple R-squared:  0.7442, Adjusted R-squared:  0.7426
## F-statistic: 452.4 on 6 and 933 DF,  p-value: < 2.2e-16
```

From this two models, we get the knowledge that only KDA and GPM are important for a match. Now, we have confidence to eliminate duration and XPM from the model:

```
dota.lm.radiant.sum.reduced <- lm(as.numeric(Winner)~Dradiant+Aradiant+Gradiant,dota.sum)
summary(dota.lm.radiant.sum.reduced)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ Dradiant + Aradiant + Gradiant,
##     data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94318 -0.17522  0.00061  0.17770  0.77555
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.312e-01  5.993e-02   12.20   <2e-16 ***
## Dradiant    -1.683e-02  7.228e-04  -23.29   <2e-16 ***
## Aradiant     4.666e-03  4.053e-04   11.51   <2e-16 ***
## Gradiant     4.898e-04  3.099e-05   15.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2602 on 936 degrees of freedom
## Multiple R-squared:  0.7303, Adjusted R-squared:  0.7294
## F-statistic: 844.8 on 3 and 936 DF,  p-value: < 2.2e-16
```

```
dota.lm.dire.sum.reduced <- lm(as.numeric(Winner)~Ddire+Adire+Gdire,dota.sum)
summary(dota.lm.dire.sum.reduced)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ Ddire + Adire + Gdire, data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14231 -0.18271 -0.00451  0.17207  0.69035
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.365e+00  5.246e-02   45.09   <2e-16 ***
## Ddire        1.568e-02  6.999e-04   22.41   <2e-16 ***
## Adire       -4.743e-03  3.725e-04  -12.73   <2e-16 ***
## Gdire       -5.111e-04  2.734e-05  -18.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.2548 on 936 degrees of freedom
## Multiple R-squared:  0.7414, Adjusted R-squared:  0.7405
## F-statistic: 894.3 on 3 and 936 DF,  p-value: < 2.2e-16
```

```
cbind(dota.lm.radiant.sum.reduced$coefficients,dota.lm.dire.sum.reduced$coefficients)
```

```
##                       [,1]          [,2]
## (Intercept)  0.7312274553  2.3653101343
## Dradiant    -0.0168302406  0.0156843651
## Aradiant     0.0046659680 -0.0047433077
## Gradiant     0.0004898199 -0.0005110709
```

Final logistic regression model

In the final model for either radiant or dire, we can see the most significant factors are Death, Assistance and GPM. The coefficients are opposite number pairs, which indicates that the result of the game are very fair.

Further, we are curious about whay if we put the factors together for both side.

```
dota.lm <- lm(as.numeric(Winner)~Kradiant+Kdire+Dradiant+Ddire+Aradiant+Adire+Gradiant+Gdire,dota.sum)
summary(dota.lm)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ Kradiant + Kdire + Dradiant +
##     Ddire + Aradiant + Adire + Gradiant + Gdire, data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90038 -0.15467 -0.00072  0.15205  0.63811
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.556e+00  7.325e-02  21.241  < 2e-16 ***
## Kradiant     2.720e-03  8.032e-03   0.339   0.7350
## Kdire        1.651e-02  7.237e-03   2.281   0.0228 *
## Dradiant    -1.268e-02  7.296e-03  -1.738   0.0826 .
## Ddire       -4.906e-03  8.207e-03  -0.598   0.5501
## Aradiant     3.828e-03  6.798e-04   5.631 2.37e-08 ***
## Adire       -4.344e-03  6.072e-04  -7.154 1.71e-12 ***
## Gradiant     4.447e-04  3.057e-05  14.543  < 2e-16 ***
## Gdire       -4.685e-04  2.914e-05 -16.081  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2265 on 931 degrees of freedom
## Multiple R-squared:  0.7966, Adjusted R-squared:  0.7949
## F-statistic: 455.8 on 8 and 931 DF,  p-value: < 2.2e-16
```

```
dota.lm <- lm(as.numeric(Winner)~Dradiant+Ddire+Aradiant+Adire+Gradiant+Gdire,dota.sum)
summary(dota.lm)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ Dradiant + Ddire + Aradiant +
##     Adire + Gradiant + Gdire, data = dota.sum)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.88136 -0.15254 -0.00191  0.15272  0.63135
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.557e+00  7.337e-02  21.217  < 2e-16 ***
## Dradiant     3.424e-03  1.696e-03   2.020   0.0437 *
## Ddire       -2.882e-03  1.798e-03  -1.603   0.1093
## Aradiant     3.990e-03  6.767e-04   5.896 5.21e-09 ***
## Adire       -4.270e-03  6.040e-04  -7.069 3.06e-12 ***
## Gradiant     4.457e-04  3.062e-05  14.555  < 2e-16 ***
## Gdire       -4.692e-04  2.918e-05 -16.079  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2269 on 933 degrees of freedom
## Multiple R-squared:  0.7955, Adjusted R-squared:  0.7942
## F-statistic: 604.8 on 6 and 933 DF,  p-value: < 2.2e-16
```

```
dota.lm <- lm(as.numeric(Winner)~Aradiant+Adire+Gradiant+Gdire,dota.sum)
summary(dota.lm)
```

```
##
## Call:
## lm(formula = as.numeric(Winner) ~ Aradiant + Adire + Gradiant +
##     Gdire, data = dota.sum)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9053 -0.1538 -0.0047  0.1538  0.6155
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.551e+00  7.212e-02  21.506   <2e-16 ***
## Aradiant     3.227e-03  3.610e-04   8.938   <2e-16 ***
## Adire       -3.336e-03  3.490e-04  -9.559   <2e-16 ***
## Gradiant     4.194e-04  2.722e-05  15.405   <2e-16 ***
## Gdire       -4.375e-04  2.483e-05 -17.620   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2273 on 935 degrees of freedom
## Multiple R-squared:  0.7944, Adjusted R-squared:  0.7935
## F-statistic: 903.2 on 4 and 935 DF,  p-value: < 2.2e-16
```

After reduced, the linear model falls to only 4 factors: Assistance and GPM of both sides. Acutally, it can be concluded as two factors, the differences of assistance and GPM, since we have opposite numbers as the

coefficients. It is surprising to find the killing and death numbers are not so important we they are put together. This might because they have strong collinearity.

Anyway, we eventually get a model to describe the winner of the game which has an acceptable Adjusted R^2 and low p-value for all the coefficients. F-test also shows the model does well in total. The model can explain more than 79% of the variance. We can estimate this model, when extended to larger samples, will probably get a high accuracy.

Conclusion

The model we get from logistic regression fits our expectation on somewhere and also brings us some surprise. The assistance looks like more important than the killing and death. This result, in some extent, suggests the players join combat more to assistant their teammates and farm more money for the whole team. In the game, it is hard to do both of farming money and joining the battle. The team who can get the balance of this two factors will have higer probability to win the game.

Also, the intercept of the model shows that the game is quite fair for both sides, because the intercepts is almost at the center of 1 and 2 which presents the numeric position for Radiant and Dire.

The model still has some inaccurate part. What we consider next is how they conform these factors. Do they have inner connections or public components in different fators? We can use Principal Components Analysis to get more information.

2. Principal Components Analysis

In PCA process, we care more about what are the main factors that influence the data. It is probably separated into two sets: Radiant and Dire. PCA is good for us to comprehense how a factor weights in the model.

PCA for all factors

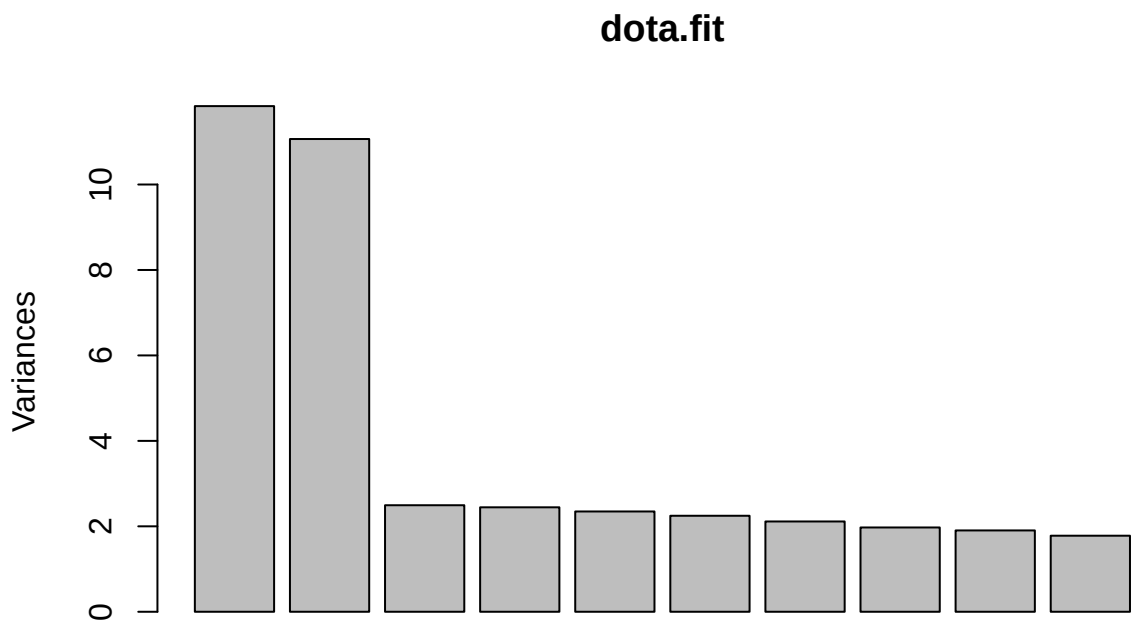
First, we apply the PCA for all numbers in dota.parameters to get a general impression of the data.

```
library(psych)
dota.fit <- prcomp(dota.parameters[,-1], retx=TRUE, center=TRUE, scale.=TRUE)
summary(dota.fit)
```

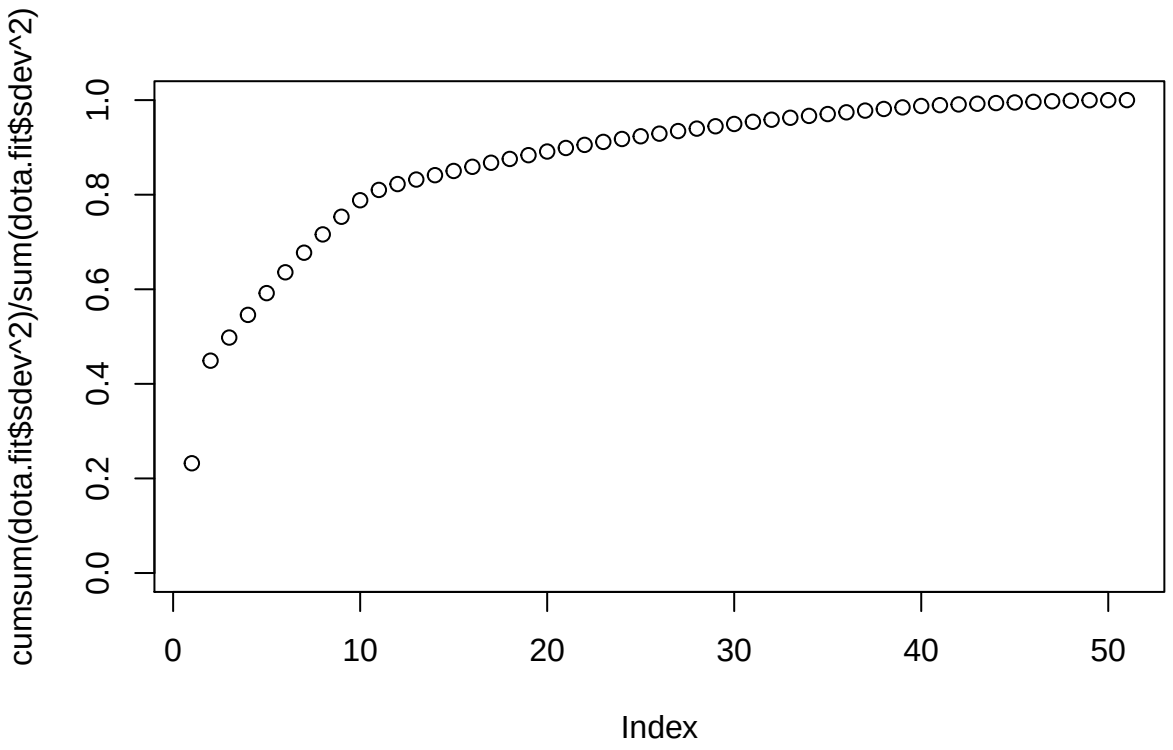
```
## Importance of components:
##                           PC1   PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.4402 3.326 1.5791 1.56352 1.53211 1.49888 1.45372
## Proportion of Variance 0.2321 0.217 0.0489 0.04793 0.04603 0.04405 0.04144
## Cumulative Proportion  0.2321 0.449 0.4979 0.54585 0.59187 0.63592 0.67736
##                           PC8     PC9   PC10    PC11    PC12    PC13
## Standard deviation     1.40453 1.37963 1.3341 1.05526 0.79469 0.69737
## Proportion of Variance 0.03868 0.03732 0.0349 0.02183 0.01238 0.00954
## Cumulative Proportion  0.71604 0.75336 0.7883 0.81010 0.82248 0.83202
##                           PC14    PC15    PC16    PC17    PC18    PC19
## Standard deviation     0.68651 0.67574 0.66866 0.66046 0.64181 0.63807
## Proportion of Variance 0.00924 0.00895 0.00877 0.00855 0.00808 0.00798
## Cumulative Proportion  0.84126 0.85021 0.85898 0.86753 0.87561 0.88359
##                           PC20    PC21    PC22    PC23    PC24    PC25
## Standard deviation     0.62621 0.61806 0.57848 0.56831 0.56031 0.53967
## Proportion of Variance 0.00769 0.00749 0.00656 0.00633 0.00616 0.00571
```

```
## Cumulative Proportion  0.89128 0.89877 0.90533 0.91166 0.91782 0.92353
##                           PC26    PC27    PC28    PC29    PC30    PC31
## Standard deviation     0.53218 0.52763 0.51533 0.50178 0.49905 0.48117
## Proportion of Variance 0.00555 0.00546 0.00521 0.00494 0.00488 0.00454
## Cumulative Proportion  0.92908 0.93454 0.93975 0.94469 0.94957 0.95411
##                           PC32    PC33    PC34    PC35    PC36    PC37
## Standard deviation     0.47569 0.46106 0.44804 0.44437 0.43344 0.43103
## Proportion of Variance 0.00444 0.00417 0.00394 0.00387 0.00368 0.00364
## Cumulative Proportion  0.95855 0.96272 0.96665 0.97052 0.97421 0.97785
##                           PC38    PC39    PC40    PC41    PC42    PC43
## Standard deviation     0.41760 0.41448 0.38743 0.30341 0.27823 0.27131
## Proportion of Variance 0.00342 0.00337 0.00294 0.00181 0.00152 0.00144
## Cumulative Proportion  0.98127 0.98464 0.98758 0.98939 0.99090 0.99235
##                           PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.2671 0.25977 0.25358 0.24896 0.24291 0.23561
## Proportion of Variance 0.0014 0.00132 0.00126 0.00122 0.00116 0.00109
## Cumulative Proportion  0.9938 0.99507 0.99633 0.99755 0.99870 0.99979
##                           PC50    PC51
## Standard deviation     0.07651 0.06937
## Proportion of Variance 0.00011 0.00009
## Cumulative Proportion  0.99991 1.00000
```

```
screeplot(dota.fit)
```



```r
plot(cumsum(dota.fit$sdev^2)/sum(dota.fit$sdev^2),ylim=c(0,1))
```



As expected, the first two components are large and share close weight. It indicates that there are important factors can influence the result strongly. Also, the screeplot and the summary show the variance value of the result. from which we can see the first 11 components explains 80% of the variance we want to retain.

For the same reason, the separated elements of heroes skill statistics cannot behave a balance model in all. We are curious about what the sum say in the data.

PCA for sums

```r
dota.fit.sum <- prcomp(dota.sum[,-1], retx=TRUE, center=TRUE, scale.=TRUE)
dota.fit.sum
```

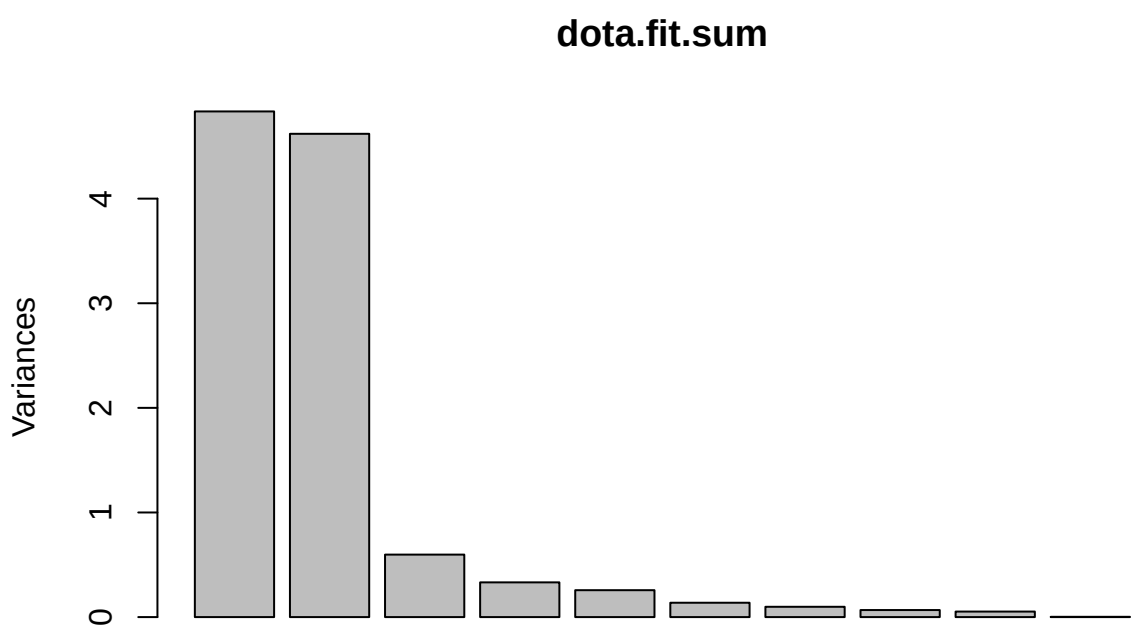
```
## Standard deviations:
##  [1] 2.19854286 2.14931547 0.77274864 0.57619354 0.50725405 0.37026383
##  [7] 0.31358856 0.25956768 0.22986338 0.05069199 0.04680645
##
## Rotation:
##                 PC1         PC2         PC3         PC4          PC5
## duration  0.1770165 -0.34429066 -0.31991279  0.84377756 -0.004304666
## Kradiant -0.1210998 -0.43656535 -0.07820759 -0.21147409  0.228499606
## Dradiant  0.4060622 -0.18122489 -0.06815799 -0.26606621 -0.187938696
## Aradiant -0.1033753 -0.42581671 -0.18592643 -0.09718973  0.356763988
```

```
## Xradiant -0.1950360 -0.36410069  0.49134691  0.07287626 -0.166746843
## Gradiant -0.2932018 -0.28745897  0.36789287  0.08812806 -0.518717302
## Kdire     0.4087741 -0.17488052 -0.07149599 -0.24678430 -0.197049646
## Ddire    -0.1134160 -0.43841297 -0.07320500 -0.24074537  0.228810443
## Adire     0.3860106 -0.17879760 -0.22098080 -0.10874627 -0.440504660
## Xdire     0.3994869 -0.08105241  0.50259854  0.10932891  0.152248571
## Gdire     0.4072680  0.01353238  0.40453655  0.08581481  0.424133221
##                 PC6          PC7          PC8         PC9          PC10
## duration -0.04614128  0.176520209 -0.007023268  0.04499484 -0.0090152046
## Kradiant -0.03237937  0.243818772  0.372353746 -0.08083013 -0.0534311486
## Dradiant -0.05730428  0.282976633 -0.285950397  0.12119918 -0.7128624711
## Aradiant  0.20089029 -0.530529261 -0.549687146 -0.10913348 -0.0173320937
## Xradiant -0.47250381 -0.234985174 -0.018507553  0.52533779 -0.0073468875
## Gradiant  0.56003473  0.172672785 -0.110692651 -0.25200439 -0.0003822309
## Kdire    -0.06395752  0.275603532 -0.335233683  0.13265530  0.6921433073
## Ddire    -0.02677872  0.236041944  0.322085832 -0.07665220  0.0961188525
## Adire     0.12151241 -0.569871438  0.473081215 -0.02741448  0.0134577497
## Xdire    -0.31462457 -0.079956031 -0.009002366 -0.66379822  0.0034904912
## Gdire     0.54563868 -0.001580704  0.157405836  0.40062946  0.0064749661
##                  PC11
## duration -0.0190335100
## Kradiant  0.6951000098
## Dradiant -0.0593632990
## Aradiant  0.0226707910
## Xradiant  0.0004659391
## Gradiant  0.0025805916
## Kdire     0.0915886808
## Ddire    -0.7099103989
## Adire    -0.0082086817
## Xdire    -0.0002550337
## Gdire     0.0003885424
```

```
summary(dota.fit.sum)
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.1985 2.1493 0.77275 0.57619 0.50725 0.37026
## Proportion of Variance 0.4394 0.4200 0.05429 0.03018 0.02339 0.01246
## Cumulative Proportion  0.4394 0.8594 0.91366 0.94384 0.96724 0.97970
##                            PC7     PC8    PC9    PC10    PC11
## Standard deviation     0.31359 0.25957 0.2299 0.05069 0.04681
## Proportion of Variance 0.00894 0.00613 0.0048 0.00023 0.00020
## Cumulative Proportion  0.98864 0.99476 0.9996 0.99980 1.00000
```

```
screeplot(dota.fit.sum)
```



Now, the sums do a great improvement to the model that first two components explain more than 85% of the variance. It is a very good sign when we want to reduce the factors. Also, we notice that this two components have close standard deviation and proportion of variance. It is like what we predicted before the test, they are probably the performance of Radiant and Dire teams.

To prove this view, we can make a comparison of the coefficients of the two components.

```
sort(dota.fit.sum$rotation[,1])
```

```
##   Gradiant   Xradiant   Kradiant      Ddire   Aradiant   duration
## -0.2932018 -0.1950360 -0.1210998 -0.1134160 -0.1033753  0.1770165
##      Adire      Xdire   Dradiant      Gdire      Kdire
##  0.3860106  0.3994869  0.4060622  0.4072680  0.4087741
```

```
sort(dota.fit.sum$rotation[,2])
```

```
##       Ddire    Kradiant    Aradiant    Xradiant    duration    Gradiant
## -0.43841297 -0.43656535 -0.42581671 -0.36410069 -0.34429066 -0.28745897
##    Dradiant       Adire       Kdire       Xdire       Gdire
## -0.18122489 -0.17879760 -0.17488052 -0.08105241  0.01353238
```

```
sort(abs(dota.fit.sum$rotation[,1]))
```

```
##  Aradiant     Ddire  Kradiant  duration  Xradiant  Gradiant     Adire
## 0.1033753 0.1134160 0.1210998 0.1770165 0.1950360 0.2932018 0.3860106
##     Xdire  Dradiant     Gdire     Kdire
## 0.3994869 0.4060622 0.4072680 0.4087741
```

```r
sort(abs(dota.fit.sum$rotation[,2]))
```

```
##      Gdire      Xdire      Kdire      Adire   Dradiant   Gradiant
## 0.01353238 0.08105241 0.17488052 0.17879760 0.18122489 0.28745897
##   duration   Xradiant   Aradiant   Kradiant      Ddire
## 0.34429066 0.36410069 0.42581671 0.43656535 0.43841297
```

From the sort of the abstract value of the rotation, the order of these factors is very interesting. We can see similar orders of PC1 and opposite PC2 and close values for Radiant in PC1 against Dire in PC2. It means they are orthogonal and function importantly.

From my own guess:

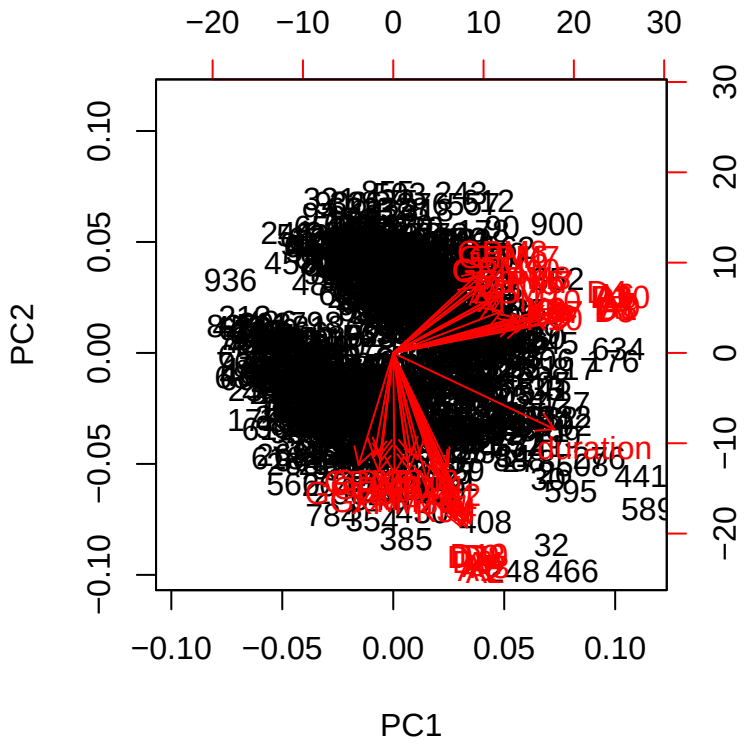
PC1 is mostly influenced by killings and assistance in the game. It reflects how a team performanced in the match which can be explained as their positiveness in this game.

PC2, on the opposite, is almost negative estimates, and the largest coefficients are deaths. This component shows their opponent's performance, or their mistakes in the game. So, I call this factor negativeness of the team.

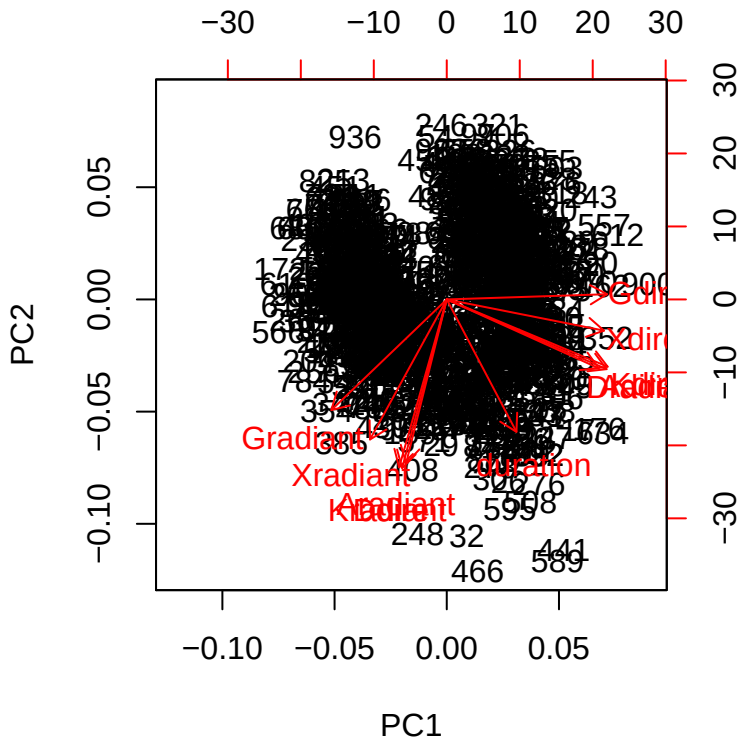
This discovery is helpful. It removes out other noise from the match. If we use the coefficients to weight of these numbers for a team, we can easily tell them how should perform to get a balance strategy in game and how can they beat their opponents by analyzing enemies' skill characteristics.

Biplot

```r
biplot(dota.fit)
```




```
biplot(dota.fit.sum)
```



In the biplot, it is very clear to see the orthogonality of PC1 and PC2. Another interesting thing is the duration, no matter in dota.fit or dota.fit.sum, it locates on the center of the two components. It is a indirect proof for our guess that each PC1 and PC2 presents one side of the game.

Conclusion

PCA analysis is not a clear supervised learning, so we cannot give a certain conclusion of what the components are. But we can still dig out some data traits from all factors. When we get the ingrediants of the main components, the next step is to compose the principal components with the coefficients. With them, it is easy to analyze the teams and players' behavior. And it can be a criteria to guide them.

3. Cluster Analysis

K-means

First use k-means and examine the centers.

Since we have two results of the match, it is reasonable to set k=2 for clustering.

```
library(cluster)
k<-2
dota.k<-kmeans(dota.parameters[,-1],k)
dota.k$centers
```

```
##   duration       K1       D1        A1     XPM1     GPM1       K2       D2
```

```
## 1 33.73474 6.433684 3.945263 14.290526 445.8358 454.1074 6.614737 3.526316
## 2 35.56989 3.933333 7.324731  8.273118 341.3957 327.5075 4.027957 7.017204
##          A2     XPM2     GPM2       K3       D3        A3     XPM3
## 1 14.412632 447.1011 460.6463 6.983158 3.688421 14.115789 461.6905
## 2  8.283871 353.5634 337.7118 3.763441 7.126882  8.348387 336.2731
##       GPM3       K4       D4       A4     XPM4     GPM4       K5       D5
## 1 471.4126 6.616842 3.656842 13.59158 449.5747 465.9432 6.162105 3.730526
## 2 328.8839 3.845161 7.126882  8.27957 337.6000 326.0387 3.741935 6.879570
##          A5     XPM5     GPM5       K6       D6        A6     XPM6
## 1 14.000000 440.5095 445.9642 3.391579 6.964211  7.757895 325.9705
## 2  8.137634 344.5011 335.7763 7.002151 4.008602 15.219355 464.9699
##       GPM6       K7       D7        A7     XPM7     GPM7       K8       D8
## 1 316.6021 3.823158 6.593684  7.528421 345.3074 336.6695 3.701053 6.663158
## 2 470.1075 8.075269 3.875269 14.395699 491.5699 499.3978 6.720430 4.184946
##          A8     XPM8     GPM8       K9       D9        A9     XPM9
## 1  7.456842 338.4653 329.8526 3.351579 6.547368  7.545263 330.4632
## 2 15.021505 462.4473 464.0817 6.346237 4.045161 14.987097 444.6839
##       GPM9      K10      D10       A10    XPM10    GPM10
## 1 323.2084 3.498947 6.661053  7.505263 334.0505 321.6400
## 2 448.4129 6.531183 4.070968 15.490323 447.6516 451.8151
```

As above, these centers show symetricity for Radiant and Dire if we compare K1 and K6, D1 and D6 etc. in the two clusters.

We can compare the k-means clustering with the winner of the game, the result can shows that the clustering in some extent present who wins the game.

```
table(dota.parameters$Winner,dota.k$cluster)
```

```
##
##             1   2
##   Dire      6 455
##   Radiant 469  10
```

The k-means clustering get a (455+469)/966=95.65% correct rate. However, it is not that useful if we recognize it is a review of the game rather than a prediction. If we get the stats of the game, it is not meaningful to infer the result from it.

What we care most here is the prediction of the game. It is rather tough to achieve. A further discussion will be illustrated in SVM model.

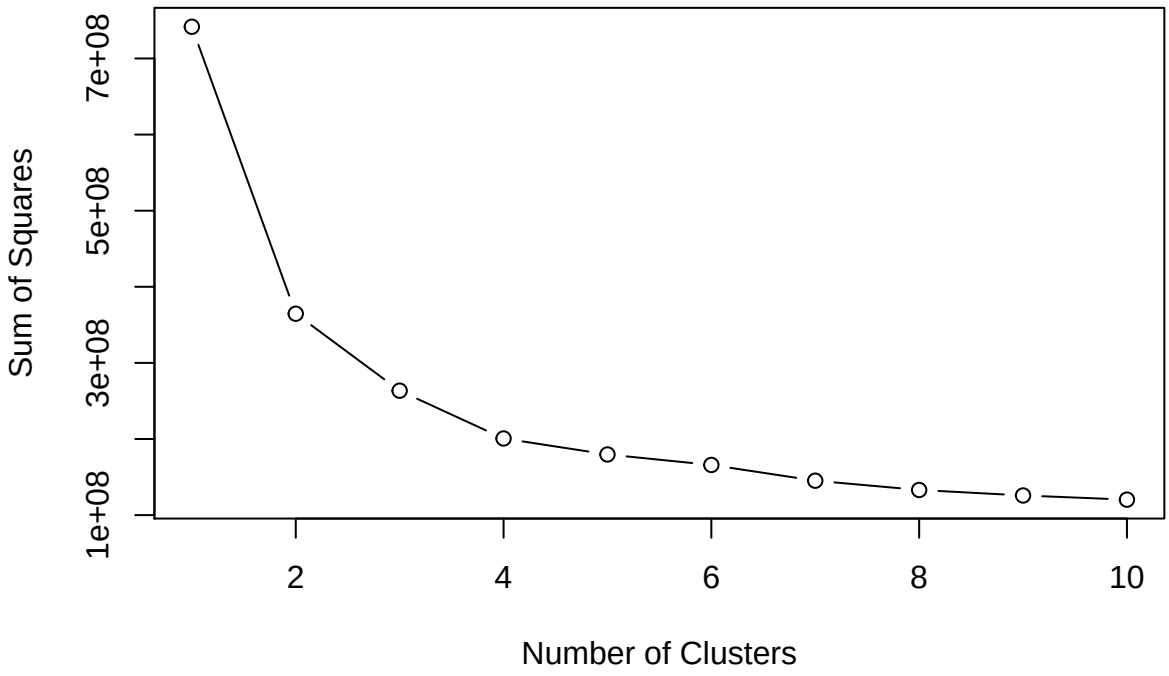
```
sos <- (nrow(dota.sum[,-1])-1)*sum(apply(dota.sum[,-1],2,var))
for (i in 2:10) sos[i] <- sum(kmeans(dota.sum[,-1], centers=i)$withinss)
plot(1:10, sos, type="b", xlab="Number of Clusters", ylab="Sum of Squares")
library(useful)
```

```
## Warning: package 'useful' was built under R version 3.2.4
```

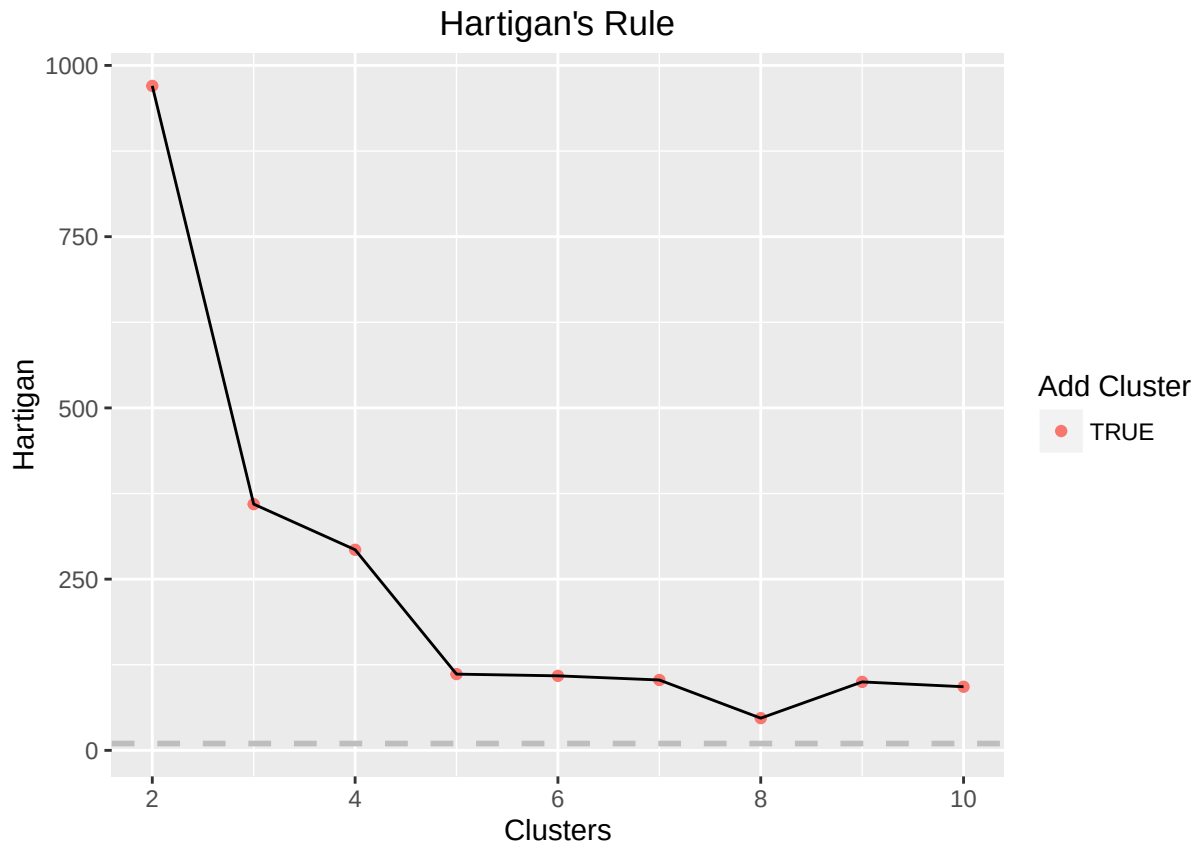
```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```




```
best<-FitKMeans(dota.sum[,-1],max.clusters=10, seed=111)
PlotHartigan(best)
```




```
k<-5
dota.k.sum<-kmeans(dota.sum[,-1],k)
dota.k.sum$centers
```

```
##   duration  Kradiant Dradiant Aradiant Xradiant Gradiant     Kdire
## 1 38.43220 20.966102 36.54661 45.14831 1785.860 1734.691 35.627119
## 2 46.66667 37.184211 42.33333 77.87719 2364.763 2147.570 41.254386
## 3 25.17327 27.584158 10.13366 57.34653 2053.946 2236.639  9.529703
## 4 25.56376  8.791946 29.06711 19.04698 1296.698 1323.913 28.536913
## 5 38.82845 35.548117 22.91632 78.26360 2358.941 2336.038 22.071130
##       Ddire    Adire    Xdire    Gdire
## 1 21.944915 78.49576 2369.047 2373.805
## 2 38.122807 87.43860 2547.798 2355.596
## 3 28.069307 20.44059 1340.634 1359.183
## 4  9.409396 59.93289 2055.913 2212.685
## 5 36.267782 47.39749 1855.038 1780.377
```

But we can use larger k for clustering. The Hartigan graph shows that using k=5 is a good number for clustering. Though the result is hard to explicitly explain, we can still name them as short match(one side overwhelming) and long match(well-matched). Based on the duration centers, we can conclude the 25 minutes clusters are short games winned by both sides; 38, longer matches and 46, longest games. It indicates if the game duration is over 46 minutes, it is hard to predict who wins.

Conclusion

The Clustering analysis indicates that the game result can be predicted by technical statistics. If we gather player's match data and use an algorithm to get his average performance in a match, we can generate predictions before the game.

4. Elastic Net Model

Since we do not only care the winner, but also the duration. It is meaningful to see how can the stats of teams determine the duration. Here, we can use lasso and ridge regression for the test.

Actually, the XPM and GPM cannot affect the duration strongly, since they are already diveded by time(per minute). And the kill number of Dire equals the death number of radiant. So we can remove the Kdire, Ddire and XPM and GPM for both sides.

Set up the train and test sample

```
set.seed(1013)
train <- sample(1:nrow(dota.sum),nrow(dota.sum)/2)
test <- (-train)
in_sample <- dota.sum[train,]
out_sample <- dota.sum[test,]
trainx <- as.matrix(in_sample[,c(3:5,10)])
trainy <- in_sample$duration
testx <- as.matrix(out_sample[,c(3:5,10)])
testy <- out_sample$duration
```

Lasso and ridge regression

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.5

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.2.4

##
## Attaching package: 'foreach'

## The following object is masked from 'package:chron':
##
##     times

## Loaded glmnet 2.0-5
```

```
lambdalevels <- 10^seq(7,-2,length=100)
dota.cv.mod<-NULL
bestlambda<-NULL
yhat<-NULL
mse<-NULL
i<-0
for(a in seq(0,1,0.1)){
  i<-i+1
  dota.cv.mod[[i]]=cv.glmnet(trainx,trainy,alpha=a,lambda=lambdalevels)
  bestlambda[i] <- dota.cv.mod[[i]]$lambda.min
  yhat[[i]] <- predict(dota.cv.mod[[i]]$glmnet.fit, s=dota.cv.mod[[i]]$lambda.min, newx=trainx)
  mse[i] <- sum((trainy - yhat[[i]])^2)/nrow(trainx)
}
mse
```

```
##  [1] 59.99502 59.90998 59.93437 59.89970 59.93392 59.89978 59.90927
##  [8] 59.90232 59.88802 59.88434 59.88440
```

```
best<-order(mse)[1]
best
```

```
## [1] 10
```

```
bestalpha<-(best-1)*0.1
bestalpha
```

```
## [1] 0.9
```

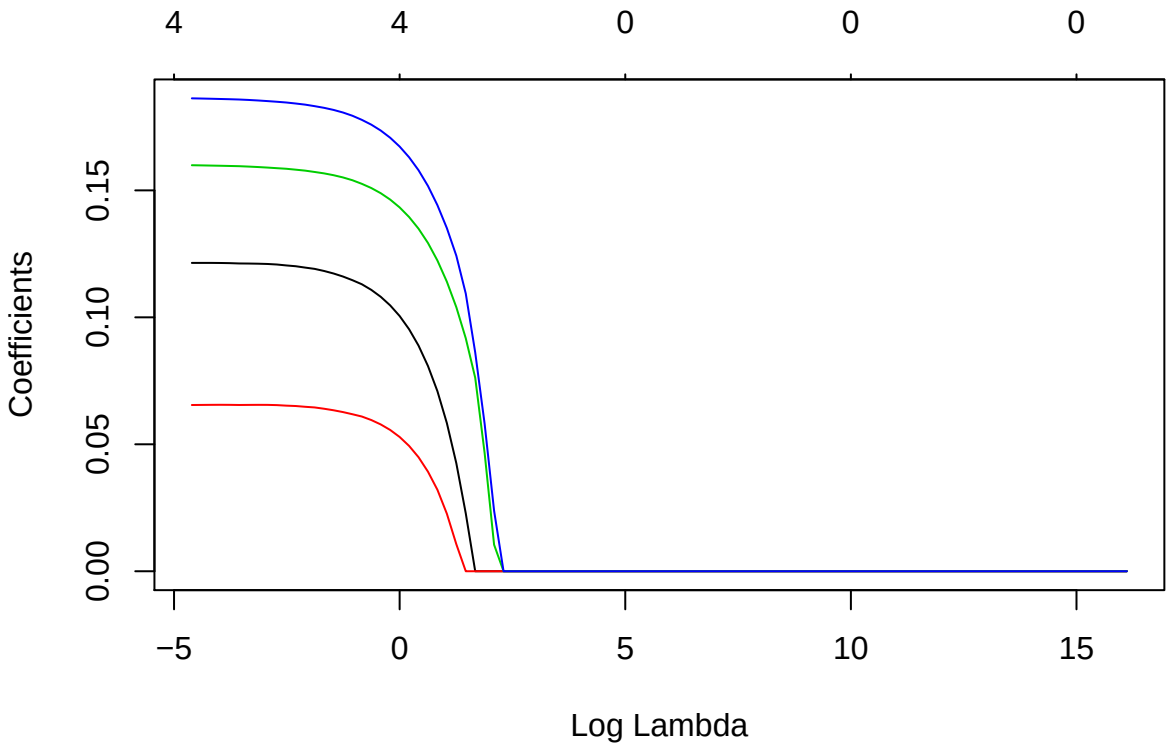
```
bestlambdafinal<-bestlambda[best]
bestlambdafinal
```

```
## [1] 0.01
```

```
#so, the best alpha is 1, best lambda is 0.01
mod=glmnet(trainx,trainy,alpha=bestalpha,lambda=lambdalevels)
coef(mod)[,100]
```

```
## (Intercept)    Kradiant    Dradiant    Aradiant       Adire
##  9.92794291  0.12142973  0.06546708  0.15990349  0.18624061
```

```
plot(mod,xvar="lambda")
```



We can use R^2 to evaluate the model:

```
yhat.test <- predict(dota.cv.mod[[best]]$glmnet.fit, s=bestlambdafinal, newx=testx)
mse.test <- sum((testy - yhat.test)^2)/nrow(testx)
mse.test
```

```
## [1] 64.81355
```

```
tss <- sum((testy - mean(testy))^2)
sse <- sum((testy - yhat.test)^2)
r2 <- (tss - sse) / tss
r2
```

```
## [1] 0.602968
```

This result is acceptable, but not very good in predicting the duration. It explains about 60% of the variance here.

The coefficients are shown as below:

```
predict(mod, type="coefficients",s=bestlambdafinal)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       1
```

```
## (Intercept) 9.92794291
## Kradiant    0.12142973
## Dradiant    0.06546708
## Aradiant    0.15990349
## Adire       0.18624061
```

Conclusion

It seems the game will always last longer than 10 minutes and can be predicted by the skill stats.

With the increasing of the time, according to the model, the death number and assistance number grow gradually. It, in some extent, is a indicator in evaluating a team's performance in future.

5. Support Vector Machine

Set up the train and test sample

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.4
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

## The following object is masked from 'package:psych':
##
##     alpha
```

```
trainset <- dota.sum[train,]
trainx <- dota.sum[train,-1]
trainy <- dota.sum[train,1]
testx <- dota.sum[test,-1]
testy <- dota.sum[test,1]
```

Choosing kernel

Radial kernel

```
costvalues <- 10^seq(-3,2,1)
tuned.svm <- tune(svm, Winner~., data=trainset, ranges=list(cost=costvalues), kernel="radial")
summary(tuned.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01276596
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.55531915 0.04423395
## 2 1e-02 0.04042553 0.03393897
## 3 1e-01 0.03191489 0.04043175
## 4 1e+00 0.01276596 0.02055514
## 5 1e+01 0.02127660 0.01737227
## 6 1e+02 0.01914894 0.01862968
```

```
#the best performance is when we set cost as 1, we get 5% of the points incorrectly classified.
svmfit <- svm(Winner~. , kernel = "radial",cost = 100,data=trainset)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Winner ~ ., data = trainset, kernel = "radial",
##     cost = 100)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##       gamma:  0.09090909
##
## Number of Support Vectors:  32
##
##  ( 16 16 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dire Radiant
```

Linear kernel

```
tuned.svm <- tune(svm, Winner~., data=trainset, ranges=list(cost=costvalues), kernel="linear")
summary(tuned.svm)
```

```
##
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.0106383
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.02978723 0.02497421
## 2 1e-02 0.01914894 0.01569925
## 3 1e-01 0.01276596 0.01487672
## 4 1e+00 0.01063830 0.01121375
## 5 1e+01 0.01489362 0.01436061
## 6 1e+02 0.02127660 0.01418440
```

```
#the best cost is 1, where we get only 4.2% is wrong
svmfit <- svm(Winner~. , kernel = "linear",cost = 1,data=trainset)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Winner ~ ., data = trainset, kernel = "linear",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.09090909
##
## Number of Support Vectors:  27
##
##  ( 13 14 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dire Radiant
```

Polynomial kernel

```
tuned.svm <- tune(svm, Winner~., data=trainset, ranges=list(cost=costvalues), kernel="polynomial")
summary(tuned.svm)
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.01276596
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.53191489 0.05673759
## 2 1e-02 0.05744681 0.03179647
## 3 1e-01 0.02127660 0.02653658
## 4 1e+00 0.01914894 0.02115806
## 5 1e+01 0.01276596 0.02055514
## 6 1e+02 0.01702128 0.02197437
```

```
#the best cost is 100, where we get only 4.2% is wrong
svmfit <- svm(Winner~. , kernel = "polynomial",cost = 100,data=trainset)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Winner ~ ., data = trainset, kernel = "polynomial",
##     cost = 100)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  100
##      degree:  3
##       gamma:  0.09090909
##      coef.0:  0
##
## Number of Support Vectors:  31
##
##  ( 18 13 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dire Radiant
```

Sigmoid kernel

```
tuned.svm <- tune(svm, Winner~., data=trainset, ranges=list(cost=costvalues), kernel="sigmoid")
summary(tuned.svm)
```

```
##
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.02340426
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.59148936 0.05094549
## 2 1e-02 0.03191489 0.02700629
## 3 1e-01 0.02340426 0.02115806
## 4 1e+00 0.03617021 0.02845731
## 5 1e+01 0.06382979 0.03884557
## 6 1e+02 0.06808511 0.03725936
```

```
#the best cost is 0.1, where we get only 4.2% is wrong
svmfit <- svm(Winner~. , kernel = "sigmoid",cost = 0.1,data=trainset)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Winner ~ ., data = trainset, kernel = "sigmoid",
##     cost = 0.1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  0.1
##       gamma:  0.09090909
##      coef.0:  0
##
## Number of Support Vectors:  132
##
##  ( 66 66 )
##
##
## Number of Classes:  2
##
## Levels:
##  Dire Radiant
```

Prediction for test samples

It seems that the linear kernel gives the best result. So, we use linear kernel to test the out-sample.

```
prediction <- predict(tuned.svm$best.model,testx)
table(prediction,truth=testy)
```

```
##           truth
## prediction Dire Radiant
##    Dire     216       4
##    Radiant    9     241
```

```
sum(yhat==testy)/length(testy)
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## Warning in `==.default`(yhat, testy): longer object length is not a
## multiple of shorter object length
```

```
## [1] 0
```

The SVM is satifactory, a high correct rate more than 97% is a very excellent result in statistics.

Conclusion

In SVM model, we tested different kernals and costs for the dataset. It gives a very good response with almost 95% correct to the test data. This result may indicates that the SVM is a good tool to analyze or predict matches with enough number of samples. The statistical points of the players in dota can truely show the trend of the game. With the different method of kernel, SVM can easily do classification work in different dataset, especially when we cannot summary the trait of the data in lower dimension. Also, if we want to improve the performance, choosing the right parameters and the model is important. And maybe changing the scale of the data can help to make a better result.

6. Assosiation Rule

Set up transactions

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 3.2.4
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:kernlab':
##
##     size
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
```

```
## Warning: package 'arulesViz' was built under R version 3.2.4
```

```
## Loading required package: grid
```

```
## Warning: replacing previous import by 'utils::head' when loading
## 'arulesViz'
```

```
dota.tran <- as(dota.orgin[,c(4,5,12,19,26,33,40,47,54,61,68)],"transactions")
summary(dota.tran)
```

```
## transactions as itemMatrix in sparse format with
##  966 rows (elements/itemsets/transactions) and
##  1112 columns (items) and a density of 0.009892086
##
## most frequent items:
## Winner=Radiant    Winner=Dire     Hero9=lion  Hero3=invoker     Hero8=lion
##            494            472             51             47             46
##        (Other)
##           9516
##
## element (itemset/transaction) length distribution:
## sizes
##  11
## 966
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      11      11      11      11      11      11
##
## includes extended item information - examples:
##           labels variables  levels
## 1    Winner=Dire    Winner    Dire
## 2 Winner=Radiant    Winner Radiant
## 3  Hero1=abaddon     Hero1 abaddon
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

```
inspect(dota.tran[1:3])
```

```
##   items                         transactionID
## 1 {Winner=Dire,
##    Hero1=spirit-breaker,
##    Hero2=necrophos,
##    Hero3=enchantress,
##    Hero4=natures-prophet,
##    Hero5=phantom-lancer,
```

```
##    Hero6=bounty-hunter,
##    Hero7=invoker,
##    Hero8=lion,
##    Hero9=tidehunter,
##    Hero10=spectre}                          1
## 2 {Winner=Radiant,
##    Hero1=spectre,
##    Hero2=doom,
##    Hero3=zeus,
##    Hero4=vengeful-spirit,
##    Hero5=nyx-assassin,
##    Hero6=mirana,
##    Hero7=outworld-devourer,
##    Hero8=gyrocopter,
##    Hero9=faceless-void,
##    Hero10=lion}                             2
## 3 {Winner=Radiant,
##    Hero1=ogre-magi,
##    Hero2=outworld-devourer,
##    Hero3=zeus,
##    Hero4=enigma,
##    Hero5=batrider,
##    Hero6=weaver,
##    Hero7=alchemist,
##    Hero8=night-stalker,
##    Hero9=io,
##    Hero10=keeper-of-the-light}              3
```

```
itemFrequency(dota.tran[, 1:10])
```

```
##              Winner=Dire           Winner=Radiant            Hero1=abaddon
##              0.488612836              0.511387164              0.002070393
##          Hero1=alchemist Hero1=ancient-apparition         Hero1=anti-mage
##              0.014492754              0.006211180              0.008281573
##         Hero1=arc-warden                Hero1=axe               Hero1=bane
##              0.000000000              0.003105590              0.012422360
##           Hero1=batrider
##              0.010351967
```

Finding raw rules

It is very difficult to find rules in small sample data like this, the support is not high enough to form big set.

```
rule.dota <- apriori(dota.tran, parameter = list(support=0.005, confidence=0.7, minlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.7    0.1    1 none FALSE            TRUE   0.005      3     10
##  target   ext
##   rules FALSE
```

```
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1023 item(s), 966 transaction(s)] done [0.00s].
## sorting and recoding items ... [648 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [4 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rule.dota)
```

```
## set of 4 rules
##
## rule length distribution (lhs + rhs):sizes
## 3
## 4
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       3       3       3       3       3
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.005176   Min.   :0.8333   Min.   :1.706
##  1st Qu.:0.005176   1st Qu.:0.8333   1st Qu.:1.706
##  Median :0.005176   Median :0.9167   Median :1.876
##  Mean   :0.005176   Mean   :0.9167   Mean   :1.876
##  3rd Qu.:0.005176   3rd Qu.:1.0000   3rd Qu.:2.047
##  Max.   :0.005176   Max.   :1.0000   Max.   :2.047
##
## mining info:
##       data ntransactions support confidence
##  dota.tran           966   0.005        0.7
```

```
inspect(rule.dota)
```

```
##   lhs                                    rhs           support
## 1 {Hero3=invoker,Hero9=bounty-hunter} => {Winner=Dire} 0.005175983
## 2 {Hero6=zeus,Hero7=spectre}          => {Winner=Dire} 0.005175983
## 3 {Hero3=witch-doctor,Hero8=invoker}  => {Winner=Dire} 0.005175983
## 4 {Hero8=doom,Hero9=witch-doctor}     => {Winner=Dire} 0.005175983
##   confidence lift
## 1 0.8333333  1.705508
## 2 1.0000000  2.046610
## 3 0.8333333  1.705508
## 4 1.0000000  2.046610
```

Conclusion

These rules are very interesting and valueble. For example,

{Hero6=zeus,Hero7=spectre} => {Winner=Dire}

This rule shows that when dire get zeus and spectre together, they can win the game in a high expectation. It is a known rule for players called universal strategy since both zeus and spectre have skill can hit enemies whereever they are. It proves that the rule analysis fits the realistic strategy, thus we can use rules we do not know from the set.

If we have more samples, we can generate more rules for the game which can guide the players in a large range and help them accumalate tips against their opponents.

By decreasing the support, we can get more rules:

```
rule.dota <- apriori(dota.tran, parameter = list(support=0.003, confidence=0.7, minlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.7    0.1    1 none FALSE            TRUE   0.003      3     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 2
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1023 item(s), 966 transaction(s)] done [0.00s].
## sorting and recoding items ... [821 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.01s].
## writing ... [735 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rule.dota)
```

```
## set of 735 rules
##
## rule length distribution (lhs + rhs):sizes
##   3   4   5   6   7
## 437 144 105  42   7
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   3.691   4.000   7.000
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.003106   Min.   :0.7500   Min.   :  1.467
##  1st Qu.:0.003106   1st Qu.:1.0000   1st Qu.:  1.955
```

```
##  Median :0.003106   Median :1.0000   Median : 30.188
##  Mean   :0.003169   Mean   :0.9551   Mean   : 31.973
##  3rd Qu.:0.003106   3rd Qu.:1.0000   3rd Qu.: 48.300
##  Max.   :0.005176   Max.   :1.0000   Max.   :120.750
##
## mining info:
##       data ntransactions support confidence
##  dota.tran           966   0.003        0.7
```

```
inspect(sort(rule.dota, by = "lift")[1:10])
```

```
##    lhs                           rhs               support confidence   lift
## 1  {Hero3=batrider,
##     Hero5=spectre}           => {Hero9=mirana} 0.00310559       0.75 120.75
## 2  {Hero8=earthshaker,
##     Hero10=dark-seer}        => {Hero4=bane}   0.00310559       1.00  96.60
## 3  {Hero2=natures-prophet,
##     Hero8=earthshaker}       => {Hero4=bane}   0.00310559       1.00  96.60
## 4  {Hero8=earthshaker,
##     Hero9=outworld-devourer} => {Hero4=bane}   0.00310559       1.00  96.60
## 5  {Hero6=vengeful-spirit,
##     Hero8=earthshaker}       => {Hero4=bane}   0.00310559       1.00  96.60
## 6  {Hero7=spectre,
##     Hero10=dark-seer}        => {Hero4=bane}   0.00310559       1.00  96.60
## 7  {Hero6=vengeful-spirit,
##     Hero10=dark-seer}        => {Hero4=bane}   0.00310559       1.00  96.60
## 8  {Hero2=natures-prophet,
##     Hero9=outworld-devourer} => {Hero4=bane}   0.00310559       1.00  96.60
## 9  {Hero2=natures-prophet,
##     Hero7=spectre}           => {Hero4=bane}   0.00310559       1.00  96.60
## 10 {Hero7=spectre,
##     Hero9=outworld-devourer} => {Hero4=bane}   0.00310559       1.00  96.60
```

```
rulesdataframe<- as(rule.dota, "data.frame")
subset.matrix <- is.subset(rule.dota, rule.dota)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rule.dota[!redundant]
summary(rules.pruned)
```

```
## set of 350 rules
##
## rule length distribution (lhs + rhs):sizes
##   3
## 350
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       3       3       3       3       3
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.003106   Min.   :0.7500   Min.   : 1.467
##  1st Qu.:0.003106   1st Qu.:0.7500   1st Qu.: 1.637
```

```
##  Median :0.003106   Median :1.0000   Median : 1.955
##  Mean   :0.003233   Mean   :0.9292   Mean   : 5.734
##  3rd Qu.:0.003106   3rd Qu.:1.0000   3rd Qu.: 2.047
##  Max.   :0.005176   Max.   :1.0000   Max.   :53.667
##
## mining info:
##       data ntransactions support confidence
##  dota.tran           966   0.003        0.7
```

```
inspect(sort(rules.pruned[1:10],by = "lift"))
```

```
##    lhs                            rhs                 support confidence      lift
## 1  {Hero3=batrider,
##     Hero9=mirana}              => {Hero5=spectre}  0.00310559       1.00 48.300000
## 2  {Hero9=vengeful-spirit,
##     Hero10=templar-assassin}   => {Winner=Dire}    0.00310559       1.00  2.046610
## 3  {Hero2=invoker,
##     Hero10=templar-assassin}   => {Winner=Dire}    0.00310559       1.00  2.046610
## 4  {Hero5=death-prophet,
##     Hero7=broodmother}         => {Winner=Dire}    0.00310559       1.00  2.046610
## 5  {Hero1=zeus,
##     Hero2=rubick}              => {Winner=Dire}    0.00310559       1.00  2.046610
## 6  {Hero1=lion,
##     Hero3=ancient-apparition} => {Winner=Dire}    0.00310559       1.00  2.046610
## 7  {Hero3=invoker,
##     Hero8=omniknight}          => {Winner=Radiant} 0.00310559       1.00  1.955466
## 8  {Hero2=invoker,
##     Hero9=chen}                => {Winner=Radiant} 0.00310559       1.00  1.955466
## 9  {Hero1=doom,
##     Hero4=batrider}            => {Winner=Radiant} 0.00310559       1.00  1.955466
## 10 {Hero3=juggernaut,
##     Hero5=shadow-shaman}       => {Winner=Radiant} 0.00310559       0.75  1.466599
```

7. More Analysis on Heroes

Aside matches, we can do more analysis on heroes with these algorithms and models

Preprocessing

```
hero <- NULL
for(i in seq(5,68,7)){
heroseq <- dota.orgin[,i:(i+6)]
colnames(heroseq) <- c("Hero","K","D","A","XPM","GPM","DMG")
hero <- rbind(hero,heroseq)
}
for(i in 2:6){
  hero[,i]<-as.numeric(hero[,i])
}
```

```
## Warning: NAs introduced by coercion
```

```
damage<-sub("k","",as.character(hero$DMG))
damage <- 1000*as.numeric(damage)
```

```
## Warning: NAs introduced by coercion
```

```
hero.parameters <- cbind(hero[,-7],DMG=damage)
head(hero.parameters)
```

```
##                 Hero  K D  A XPM GPM   DMG
## 1    spirit-breaker  2 7  9 110 256  4600
## 2           spectre 18 7 20 418 661 25600
## 3         ogre-magi  0 4 18 117 223  4700
## 4  treant-protector  0 4  6  28 175  2300
## 5              doom  3 3 14 133 329  6900
## 6 outworld-devourer 14 6 10 410 522 18300
```

Win rate stats

```
result <- NULL
for(i in 1:nrow(dota.orgin)){
  if(dota.orgin[i,4]=="Dire"){
    radiant.result <- rep("lose",5)
    dire.result <- rep("win",5)
  }
  else{
    radiant.result <- rep("win",5)
    dire.result <- rep("lose",5)
  }
  result <- c(result,radiant.result,dire.result)
}

hero.win <- cbind(hero.parameters,result)

table(hero.win$Hero,hero.win$result)
```

```
##
##                       lose win
##   abaddon               23  16
##   alchemist             41  38
##   ancient-apparition    26  17
##   anti-mage             26  32
##   arc-warden             1   0
##   axe                    6   7
##   bane                  44  35
##   batrider              67  67
##   beastmaster          120 127
##   bloodseeker            3   5
##   bounty-hunter        105  97
##   brewmaster             3   4
##   bristleback           31  27
```

```
##   broodmother           25  21
##   centaur-warrunner      4   8
##   chaos-knight          27  28
##   chen                  34  23
##   clinkz                52  45
##   clockwerk             23  13
##   crystal-maiden        24  23
##   dark-seer             55  56
##   dazzle                55  46
##   death-prophet        105  92
##   disruptor             64  87
##   doom                 122 117
##   dragon-knight          9  11
##   drow-ranger           24  21
##   earth-spirit          94  90
##   earthshaker           63  83
##   elder-titan            1   4
##   ember-spirit          56  61
##   enchantress           86  69
##   enigma                31  27
##   faceless-void        121 127
##   gyrocopter            85 101
##   huskar                 7   5
##   invoker              188 188
##   io                    30  33
##   jakiro                31  27
##   juggernaut           107  95
##   keeper-of-the-light   10   5
##   kunkka                11  16
##   legion-commander      19  22
##   leshrac                9   3
##   lich                  47  32
##   lifestealer           27  18
##   lina                  60  51
##   lion                 166 191
##   lone-druid            71  72
##   luna                  13  11
##   lycan                 10  13
##   magnus                19  21
##   medusa                23  43
##   meepo                  9   8
##   mirana                18  28
##   morphling             17  36
##   naga-siren             8   4
##   natures-prophet      105  89
##   necrophos             26  30
##   night-stalker         52  50
##   nyx-assassin          25  27
##   ogre-magi             26  33
##   omniknight            44  47
##   oracle                63  78
##   outworld-devourer    104  99
##   phantom-assassin      14  16
##   phantom-lancer        55  52
```

```
##   phoenix               53  45
##   puck                  59  59
##   pudge                 28  25
##   pugna                 11  12
##   queen-of-pain         50  39
##   razor                 14   7
##   riki                   3   0
##   rubick                39  47
##   sand-king             14  19
##   shadow-demon           9   8
##   shadow-fiend           6  13
##   shadow-shaman         47  43
##   silencer              41  39
##   skywrath-mage         10  17
##   slardar               50  46
##   slark                 64  74
##   sniper                 7   7
##   spectre              112 104
##   spirit-breaker        52  63
##   storm-spirit           2   6
##   sven                 107 108
##   techies                5   6
##   templar-assassin      24  22
##   terrorblade            3   2
##   tidehunter            88  70
##   timbersaw              6  16
##   tinker                20  15
##   tiny                  16  16
##   treant-protector       7  14
##   troll-warlord          4   4
##   tusk                  31  31
##   undying               49  50
##   ursa                  65  66
##   vengeful-spirit      137 158
##   venomancer            20  22
##   viper                 38  37
##   visage                 6   5
##   warlock               18  15
##   weaver                24  27
##   windranger            73  74
##   winter-wyvern         28  20
##   witch-doctor         169 176
##   wraith-king           20  17
##   zeus                 131 118
```

```
hero.wintable <- data.frame(table(hero.win$Hero,hero.win$result))
lose <- hero.wintable[1:111,]
win <- hero.wintable[112:222,]
hero.winrate <- cbind(lose[,-2],win[,3])
hero.winrate <- cbind(hero.winrate,hero.winrate[,2]+hero.winrate[,3])
hero.winrate <- cbind(hero.winrate,hero.winrate[,3]/hero.winrate[,4])
colnames(hero.winrate) <- c("Hero","Lose","Win","Total","WinRate")

hero.winrate[order(hero.winrate$WinRate,decreasing = T),c(1,5,4)]
```

```
##                    Hero   WinRate Total
## 30          elder-titan 0.8000000     5
## 87         storm-spirit 0.7500000     8
## 93            timbersaw 0.7272727    22
## 78         shadow-fiend 0.6842105    19
## 56            morphling 0.6792453    53
## 15    centaur-warrunner 0.6666667    12
## 96     treant-protector 0.6666667    21
## 53               medusa 0.6515152    66
## 81        skywrath-mage 0.6296296    27
## 10          bloodseeker 0.6250000     8
## 55               mirana 0.6086957    46
## 42               kunkka 0.5925926    27
## 24            disruptor 0.5761589   151
## 76            sand-king 0.5757576    33
## 12           brewmaster 0.5714286     7
## 29          earthshaker 0.5684932   146
## 51                lycan 0.5652174    23
## 62            ogre-magi 0.5593220    59
## 64               oracle 0.5531915   141
## 4             anti-mage 0.5517241    58
## 26        dragon-knight 0.5500000    20
## 86       spirit-breaker 0.5478261   115
## 75               rubick 0.5465116    86
## 89              techies 0.5454545    11
## 35           gyrocopter 0.5430108   186
## 6                   axe 0.5384615    13
## 43     legion-commander 0.5365854    41
## 83                slark 0.5362319   138
## 59            necrophos 0.5357143    56
## 101    vengeful-spirit 0.5355932   295
## 48                lion 0.5350140   357
## 66     phantom-assassin 0.5333333    30
## 106              weaver 0.5294118    51
## 52               magnus 0.5250000    40
## 38                   io 0.5238095    63
## 102         venomancer 0.5238095    42
## 71               pugna 0.5217391    23
## 31         ember-spirit 0.5213675   117
## 61         nyx-assassin 0.5192308    52
## 63           omniknight 0.5164835    91
## 9           beastmaster 0.5141700   247
## 34        faceless-void 0.5120968   248
## 109        witch-doctor 0.5101449   345
## 16         chaos-knight 0.5090909    55
## 99              undying 0.5050505    99
## 21            dark-seer 0.5045045   111
## 100               ursa 0.5038168   131
## 49           lone-druid 0.5034965   143
## 107          windranger 0.5034014   147
## 88                sven 0.5023256   215
## 8              batrider 0.5000000   134
## 37             invoker 0.5000000   376
## 69                puck 0.5000000   118
```

```
## 84               sniper 0.5000000    14
## 95                 tiny 0.5000000    32
## 97         troll-warlord 0.5000000     8
## 98                 tusk 0.5000000    62
## 103               viper 0.4933333    75
## 60         night-stalker 0.4901961   102
## 25                 doom 0.4895397   239
## 20        crystal-maiden 0.4893617    47
## 28          earth-spirit 0.4891304   184
## 65    outworld-devourer 0.4876847   203
## 80              silencer 0.4875000    80
## 67        phantom-lancer 0.4859813   107
## 85               spectre 0.4814815   216
## 2              alchemist 0.4810127    79
## 11         bounty-hunter 0.4801980   202
## 82               slardar 0.4791667    96
## 90     templar-assassin 0.4782609    46
## 79         shadow-shaman 0.4777778    90
## 111                zeus 0.4738956   249
## 70                 pudge 0.4716981    53
## 54                 meepo 0.4705882    17
## 77          shadow-demon 0.4705882    17
## 40            juggernaut 0.4702970   202
## 23         death-prophet 0.4670051   197
## 27           drow-ranger 0.4666667    45
## 13           bristleback 0.4655172    58
## 33                enigma 0.4655172    58
## 39                jakiro 0.4655172    58
## 18                clinkz 0.4639175    97
## 47                  lina 0.4594595   111
## 110         wraith-king 0.4594595    37
## 68               phoenix 0.4591837    98
## 58      natures-prophet 0.4587629   194
## 50                  luna 0.4583333    24
## 14           broodmother 0.4565217    46
## 22                dazzle 0.4554455   101
## 104               visage 0.4545455    11
## 105              warlock 0.4545455    33
## 32           enchantress 0.4451613   155
## 7                   bane 0.4430380    79
## 92            tidehunter 0.4430380   158
## 72         queen-of-pain 0.4382022    89
## 94                tinker 0.4285714    35
## 36                huskar 0.4166667    12
## 108       winter-wyvern 0.4166667    48
## 1                abaddon 0.4102564    39
## 45                  lich 0.4050633    79
## 17                  chen 0.4035088    57
## 46           lifestealer 0.4000000    45
## 91           terrorblade 0.4000000     5
## 3    ancient-apparition 0.3953488    43
## 19             clockwerk 0.3611111    36
## 41  keeper-of-the-light 0.3333333    15
## 57            naga-siren 0.3333333    12
```

```
## 73                razor 0.3333333    21
## 44              leshrac 0.2500000    12
## 5            arc-warden 0.0000000     1
## 74                riki 0.0000000     3
```

```
herotemp <- hero.winrate[order(hero.winrate$WinRate,decreasing = T),c(1,5,4)]
subset(herotemp,herotemp$WinRate>=0.5&herotemp$Total>=80)
```

```
##                 Hero   WinRate Total
## 24         disruptor 0.5761589   151
## 29       earthshaker 0.5684932   146
## 64            oracle 0.5531915   141
## 86   spirit-breaker 0.5478261   115
## 75            rubick 0.5465116    86
## 35       gyrocopter 0.5430108   186
## 83             slark 0.5362319   138
## 101 vengeful-spirit 0.5355932   295
## 48              lion 0.5350140   357
## 31     ember-spirit 0.5213675   117
## 63       omniknight 0.5164835    91
## 9       beastmaster 0.5141700   247
## 34    faceless-void 0.5120968   248
## 109    witch-doctor 0.5101449   345
## 99           undying 0.5050505    99
## 21         dark-seer 0.5045045   111
## 100             ursa 0.5038168   131
## 49       lone-druid 0.5034965   143
## 107      windranger 0.5034014   147
## 88              sven 0.5023256   215
## 8           batrider 0.5000000   134
## 37           invoker 0.5000000   376
## 69              puck 0.5000000   118
```

By displaying the table, we can find the heroes with higher win rate with large number of matches. Those are the heroes fit for the current version of the game.

Linear Model

```
hero.parameters <- na.omit(hero.parameters)
hero.parameters <- subset(hero.parameters,DMG<50000)
hero.lm <- lm(as.numeric(Hero)~.,hero.parameters)
summary(hero.lm)
```

```
##
## Call:
## lm(formula = as.numeric(Hero) ~ ., data = hero.parameters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -75.068 -27.295  -2.946  28.656  69.378
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 73.2246340  1.4053193  52.105  < 2e-16 ***
## K            0.4595924  0.1414332   3.250 0.001160 **
## D           -0.8102599  0.1097028  -7.386 1.64e-13 ***
## A           -0.1898594  0.0489850  -3.876 0.000107 ***
## XPM          0.0047206  0.0023758   1.987 0.046953 *
## GPM         -0.0669091  0.0036123 -18.523  < 2e-16 ***
## DMG          0.0013246  0.0001071  12.371  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.46 on 9424 degrees of freedom
## Multiple R-squared:  0.0493, Adjusted R-squared:  0.0487
## F-statistic: 81.45 on 6 and 9424 DF,  p-value: < 2.2e-16
```

From the result, we know it is hard to regress the data with linear model, which means the stats of different heroes are very close to each other. We may want to cluster these heroes into several sets.

K-means

Since we have 5 position for a team, so we choose k=5

```
k <- 5
hero.k <- kmeans(hero.parameters[,-1],k)
hero.k$centers
```

```
##           K        D         A      XPM      GPM       DMG
## 1  1.971182 5.304336  8.093725 204.4199 295.6989  4054.511
## 2 12.307068 5.387022 15.235226 431.4009 568.2804 19835.458
## 3  4.645608 5.513176 12.324662 270.5848 398.6723  8377.162
## 4  8.198667 5.205330 13.344640 365.8552 501.9703 13461.357
## 5 17.754098 6.545082 18.778689 421.0943 597.5328 30631.557
```

```
table(hero.parameters$Hero,hero.k$cluster)
```

```
##
##                          1   2   3   4   5
##   abaddon               19   3  10   6   0
##   alchemist             11  16  18  23   4
##   ancient-apparition    19   0  20   2   0
##   anti-mage             23   7  13   9   2
##   arc-warden             0   1   0   0   0
##   axe                    6   1   3   3   0
##   bane                  56   0  18   0   0
##   batrider              42   6  65  20   0
##   beastmaster          120   4  92  26   0
##   bloodseeker            4   0   2   1   0
##   bounty-hunter         94   1  89  14   0
##   brewmaster             3   0   3   1   0
##   bristleback           10  11  17  18   1
##   broodmother           11   3  18  13   0
```

```
##   centaur-warrunner     1   0   5   4   0
##   chaos-knight         11   9  19  14   2
##   chen                 44   0  11   1   0
##   clinkz               10  24  32  26   4
##   clockwerk             7   6  13  10   0
##   crystal-maiden       25   0  15   7   0
##   dark-seer            34   4  53  15   1
##   dazzle               92   0   4   0   0
##   death-prophet        26  32  66  65   6
##   disruptor           105   0  42   1   0
##   doom                 77  13 103  41   1
##   dragon-knight         7   1   6   6   0
##   drow-ranger          24   1  15   4   0
##   earth-spirit         76   2  87  14   0
##   earthshaker          73   3  58   9   0
##   elder-titan           0   0   3   1   0
##   ember-spirit         20  27  24  32  12
##   enchantress          53   5  60  32   1
##   enigma               34   1  17   5   0
##   faceless-void       164   4  60  11   0
##   gyrocopter           23  41  45  64  12
##   huskar                2   1   4   5   0
##   invoker              53  67 124 117  10
##   io                   49   0  12   0   0
##   jakiro               31   0  18   8   0
##   juggernaut           49  29  65  54   2
##   keeper-of-the-light   7   0   7   1   0
##   kunkka               11   1  10   5   0
##   legion-commander     17   3  10  11   0
##   leshrac               4   1   4   2   0
##   lich                 24   1  33  20   0
##   lifestealer          15   9  10  10   0
##   lina                 25  13  36  34   3
##   lion                255   0  81   9   0
##   lone-druid           48  10  45  35   0
##   luna                  5   5   5   5   2
##   lycan                 7   1  10   5   0
##   magnus               19   0  18   3   0
##   medusa               10   9  30  13   1
##   meepo                 6   0   5   5   1
##   mirana               18   2  20   4   0
##   morphling             7   9  18  12   4
##   naga-siren            9   0   2   1   0
##   natures-prophet      35  14  86  53   1
##   necrophos            25   3  17  11   0
##   night-stalker        56   1  39   6   0
##   nyx-assassin         14   6  20  11   0
##   ogre-magi            36   0  18   4   0
##   omniknight           74   0   9   0   0
##   oracle               87   0  44   5   0
##   outworld-devourer    42  37  54  53  11
##   phantom-assassin     10   5   8   5   1
##   phantom-lancer       25  14  36  28   3
##   phoenix              28   9  39  18   0
```

```
##   puck                13  20  37  44   2
##   pudge               11   5  22  12   2
##   pugna                4   2  11   6   0
##   queen-of-pain        7  17  27  35   3
##   razor                3   5   7   6   0
##   riki                 2   0   0   1   0
##   rubick              53   1  27   5   0
##   sand-king           16   0  12   3   0
##   shadow-demon         9   1   5   2   0
##   shadow-fiend         4   5   2   5   3
##   shadow-shaman       55   0  26   4   0
##   silencer            27   1  35  14   1
##   skywrath-mage        5   4  12   6   0
##   slardar             50   0  35   6   0
##   slark               27  28  32  40  10
##   sniper               3   4   3   2   2
##   spectre             37  44  51  49  27
##   spirit-breaker      53   0  52   7   0
##   storm-spirit         0   1   5   1   1
##   sven                62  25  62  55   4
##   techies              2   0   3   5   0
##   templar-assassin     8  13   8  16   1
##   terrorblade          1   1   1   2   0
##   tidehunter          68   0  70  17   0
##   timbersaw            6   3   3   7   3
##   tinker               2  12   5   9   7
##   tiny                 9   9   9   5   0
##   treant-protector    19   0   0   0   0
##   troll-warlord        1   1   3   3   0
##   tusk                33   1  19   7   0
##   undying             67   0  24   5   1
##   ursa                23  26  34  35   9
##   vengeful-spirit    233   1  52   4   0
##   venomancer          10   7  11  11   3
##   viper               11   8  27  26   2
##   visage               4   1   4   2   0
##   warlock             11   4  11   6   1
##   weaver              12   8  11  18   1
##   windranger          34  28  33  43   4
##   winter-wyvern       36   0  10   2   0
##   witch-doctor       238   0  77  22   0
##   wraith-king          7   3  14  10   3
##   zeus                10  98  25  42  69
```

It is very clear with the centers of clusters that the heroes' stats can be divided into different classes which represent different positions in a team. 2 carries, 1 control and 2 supports. This classification is significant with KDA, XPM, GPM as well as Damage. All factors have obvious difference.

Future Research Discussion

The whole result of the project is acceptable. From different perspective with these model, we get some interesting analysis of the Dota 2 game. It can generate speicific strategies for teams and common players of how to select heroes and how to behave in a match.

According to the conclusion we get above, the future task of our research is the prediction of a match for two certain teams. We can separate this process into 2 questions:

(1). The prediction with certain teams before match.

(2). The prediction with certain teams after ban/pick.

The difference is whether we know the hero build of one match.

In fact, in my view, with the result we get in logistic regression, elastic net model and SVM, it is optimistic to get a model for match predicting. The next step is to collect professional player's information such as which hero he play often and what is his performance. With the average behavior of all team players, we can calculate the static score for the team. When two teams meet, we can put their past match results in a model to generate a fix coefficient for this score. And then, we use this score or the set of stats to apply our regression or SVM model. The computer will tell us who has higher probablity to win the game eventually. For my own estimation, this model may get about 60% to 65% accuracy. It is quite high in a e-sport bet.

References

1. War and Picks http://www.datdota.com/blog/?p=1323
2. Analysis of the Time Aspect of the Matches at The International 3 (Dota 2 Tournament) http://www.r-bloggers.com/analysis-of-the-time-aspect-of-the-matches-at-the-international-3-dota-2-tournament/
3. Result Prediction by Mining Replays in Dota 2 https://www.diva-portal.org/smash/get/diva2:829556/FULLTEXT01.pdf
4. Identifying Patterns in Combat that are Predictive of Success in MOBA Games http://ciigar.csc.ncsu.edu/files/bib/Yang2014-MOBASucessPatterns.pdf