Multiple regression in R

# We are applying regression to the data attributes then compare the models to structure the best way to counter how players use their Gold in the multiple game structures within # the data sets, and also to find ways to improve or limit the use of the spent gold in the games.

# Implementation:

# read directory  
setwd("D:/Desktop/regression\_R/modelling")

Define and access the data sets. # initialize the data variables

# the match data  
match\_data <- read.csv("match.csv")  
# the players data  
players\_data <- read.csv("players.csv")  
# the players ratings data  
players\_ratings <- read.csv("player\_ratings.csv")

# View the data properties

This helps us identify the structure and properties of the data.

summary(match\_data)

## match\_id start\_time duration tower\_status\_radiant  
## Min. : 0 Min. :1.447e+09 Min. : 59 Min. : 0   
## 1st Qu.:12500 1st Qu.:1.447e+09 1st Qu.: 2029 1st Qu.: 0   
## Median :25000 Median :1.448e+09 Median : 2415 Median :1536   
## Mean :25000 Mean :1.448e+09 Mean : 2476 Mean :1000   
## 3rd Qu.:37499 3rd Qu.:1.448e+09 3rd Qu.: 2872 3rd Qu.:1974   
## Max. :49999 Max. :1.448e+09 Max. :16037 Max. :2047   
## tower\_status\_dire barracks\_status\_dire barracks\_status\_radiant  
## Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 0.0 1st Qu.: 0.00 1st Qu.: 0.00   
## Median : 384.0 Median :51.00 Median :51.00   
## Mean : 935.3 Mean :34.53 Mean :34.78   
## 3rd Qu.:1972.0 3rd Qu.:63.00 3rd Qu.:63.00   
## Max. :2047.0 Max. :63.00 Max. :63.00   
## first\_blood\_time game\_mode radiant\_win negative\_votes   
## Min. : 0.00 Min. : 2.00 Length:50000 Min. : 0.00000   
## 1st Qu.: 9.00 1st Qu.:22.00 Class :character 1st Qu.: 0.00000   
## Median : 77.00 Median :22.00 Mode :character Median : 0.00000   
## Mean : 93.83 Mean :21.47 Mean : 0.01548   
## 3rd Qu.:144.00 3rd Qu.:22.00 3rd Qu.: 0.00000   
## Max. :831.00 Max. :22.00 Max. :47.00000   
## positive\_votes cluster   
## Min. : 0.00000 Min. :111.0   
## 1st Qu.: 0.00000 1st Qu.:123.0   
## Median : 0.00000 Median :133.0   
## Mean : 0.03682 Mean :142.3   
## 3rd Qu.: 0.00000 3rd Qu.:154.0   
## Max. :80.00000 Max. :242.0

summary(players\_data)

## match\_id account\_id hero\_id player\_slot   
## Min. : 0 Min. : 0 Min. : 0.00 Min. : 0   
## 1st Qu.:12500 1st Qu.: 0 1st Qu.: 21.00 1st Qu.: 2   
## Median :25000 Median : 19213 Median : 47.00 Median : 66   
## Mean :25000 Mean : 39590 Mean : 50.55 Mean : 66   
## 3rd Qu.:37499 3rd Qu.: 70638 3rd Qu.: 75.00 3rd Qu.:130   
## Max. :49999 Max. :158360 Max. :112.00 Max. :132   
##   
## gold gold\_spent gold\_per\_min xp\_per\_min   
## Min. : 0 Min. : 0 Min. : 100.0 Min. : 0   
## 1st Qu.: 586 1st Qu.: 9590 1st Qu.: 317.0 1st Qu.: 343   
## Median : 1350 Median : 13110 Median : 395.0 Median : 443   
## Mean : 1889 Mean : 14110 Mean : 415.1 Mean : 450   
## 3rd Qu.: 2742 3rd Qu.: 17635 3rd Qu.: 496.0 3rd Qu.: 550   
## Max. :46424 Max. :200000 Max. :1601.0 Max. :1559   
##   
## kills deaths assists denies   
## Min. : 0.000 Min. : 0.000 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 3.000 1st Qu.: 5.000 1st Qu.: 7.00 1st Qu.: 1.000   
## Median : 6.000 Median : 8.000 Median :11.00 Median : 3.000   
## Mean : 7.404 Mean : 7.681 Mean :11.76 Mean : 5.136   
## 3rd Qu.:10.000 3rd Qu.:10.000 3rd Qu.:16.00 3rd Qu.: 7.000   
## Max. :59.000 Max. :42.000 Max. :54.00 Max. :112.000   
##   
## last\_hits stuns hero\_damage hero\_healing   
## Min. : 0.0 Length:500000 Min. : 0 Min. : 0.0   
## 1st Qu.: 54.0 Class :character 1st Qu.: 7309 1st Qu.: 0.0   
## Median : 105.0 Mode :character Median :10946 Median : 0.0   
## Mean : 129.5 Mean :12235 Mean : 466.6   
## 3rd Qu.: 178.0 3rd Qu.:15873 3rd Qu.: 214.0   
## Max. :2000.0 Max. :74932 Max. :19222.0   
##   
## tower\_damage item\_0 item\_1 item\_2   
## Min. : 0 Min. : 0.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 118 1st Qu.: 41.00 1st Qu.: 41.00 1st Qu.: 40.00   
## Median : 554 Median : 81.00 Median : 98.00 Median : 88.00   
## Mean : 1313 Mean : 97.42 Mean : 99.28 Mean : 93.61   
## 3rd Qu.: 1750 3rd Qu.:152.00 3rd Qu.:152.00 3rd Qu.:147.00   
## Max. :15986 Max. :254.00 Max. :254.00 Max. :254.00   
##   
## item\_3 item\_4 item\_5 level   
## Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 37.00 1st Qu.: 29.00 1st Qu.: 0.00 1st Qu.:15.00   
## Median : 63.00 Median : 63.00 Median : 46.00 Median :18.00   
## Mean : 88.25 Mean : 82.84 Mean : 72.15 Mean :18.05   
## 3rd Qu.:141.00 3rd Qu.:139.00 3rd Qu.:123.00 3rd Qu.:21.00   
## Max. :254.00 Max. :254.00 Max. :254.00 Max. :25.00   
##   
## leaver\_status xp\_hero xp\_creep xp\_roshan   
## Min. :0.00000 Min. : 25 Min. : 10 Min. : 198.0   
## 1st Qu.:0.00000 1st Qu.: 5030 1st Qu.: 6129 1st Qu.: 447.0   
## Median :0.00000 Median : 8122 Median : 8653 Median : 596.0   
## Mean :0.03873 Mean : 8765 Mean : 9472 Mean : 727.5   
## 3rd Qu.:0.00000 3rd Qu.:11939 3rd Qu.:12078 3rd Qu.: 894.0   
## Max. :4.00000 Max. :29055 Max. :31642 Max. :8050.0   
## NA's :1883 NA's :68 NA's :320438   
## xp\_other gold\_other gold\_death gold\_buyback   
## Min. : 1.0 Min. : -100.0 Min. :-23757 Min. :-22602   
## 1st Qu.: 130.0 1st Qu.: 110.0 1st Qu.: -3589 1st Qu.: -1479   
## Median : 296.0 Median : 217.0 Median : -2514 Median : -1071   
## Mean : 448.6 Mean : 493.7 Mean : -2689 Mean : -1146   
## 3rd Qu.: 548.2 3rd Qu.: 390.0 3rd Qu.: -1584 3rd Qu.: -682   
## Max. :25557.0 Max. :30986.0 Max. : -29 Max. : -5   
## NA's :21036 NA's :94897 NA's :6299 NA's :352859   
## gold\_abandon gold\_sell gold\_destroying\_structure  
## Min. : 5.0 Min. : 15.0 Min. : 32   
## 1st Qu.: 333.2 1st Qu.: 137.0 1st Qu.: 680   
## Median : 764.0 Median : 350.0 Median :2754   
## Mean : 1115.7 Mean : 658.7 Mean :2449   
## 3rd Qu.: 1444.0 3rd Qu.: 819.0 3rd Qu.:3975   
## Max. :18339.0 Max. :197237.0 Max. :7610   
## NA's :479366 NA's :102593 NA's :19675   
## gold\_killing\_heros gold\_killing\_creeps gold\_killing\_roshan  
## Min. : 18 Min. : 18 Min. : 80.0   
## 1st Qu.: 3033 1st Qu.: 2105 1st Qu.: 200.0   
## Median : 4547 Median : 4176 Median : 200.0   
## Mean : 4951 Mean : 5104 Mean : 376.7   
## 3rd Qu.: 6419 3rd Qu.: 7076 3rd Qu.: 454.0   
## Max. :37658 Max. :112058 Max. :4227.0   
## NA's :1565 NA's :294 NA's :240264   
## gold\_killing\_couriers unit\_order\_none unit\_order\_move\_to\_position  
## Min. : 60 Min. :1.0 Min. : 2   
## 1st Qu.: 175 1st Qu.:1.0 1st Qu.: 2965   
## Median : 175 Median :1.0 Median : 3835   
## Mean : 197 Mean :1.2 Mean : 4095   
## 3rd Qu.: 175 3rd Qu.:1.0 3rd Qu.: 4888   
## Max. :1725 Max. :2.0 Max. :40416   
## NA's :403021 NA's :499994 NA's :42   
## unit\_order\_move\_to\_target unit\_order\_attack\_move unit\_order\_attack\_target  
## Min. : 1.00 Min. : 1.00 Min. : 1   
## 1st Qu.: 41.00 1st Qu.: 7.00 1st Qu.: 301   
## Median : 80.00 Median : 37.00 Median : 484   
## Mean : 90.21 Mean : 76.97 Mean : 609   
## 3rd Qu.: 126.00 3rd Qu.: 103.00 3rd Qu.: 767   
## Max. :2427.00 Max. :2919.00 Max. :12866   
## NA's :54550 NA's :113271 NA's :105   
## unit\_order\_cast\_position unit\_order\_cast\_target unit\_order\_cast\_target\_tree  
## Min. : 1.00 Min. : 1.00 Min. : 1.00   
## 1st Qu.: 18.00 1st Qu.: 23.00 1st Qu.: 3.00   
## Median : 46.00 Median : 51.00 Median : 4.00   
## Mean : 62.96 Mean : 68.41 Mean : 5.04   
## 3rd Qu.: 85.00 3rd Qu.: 91.00 3rd Qu.: 5.00   
## Max. :2822.00 Max. :4241.00 Max. :270.00   
## NA's :2269 NA's :16674 NA's :35712   
## unit\_order\_cast\_no\_target unit\_order\_cast\_toggle unit\_order\_hold\_position  
## Min. : 1.0 Min. : 1.0 Min. : 1.0   
## 1st Qu.: 43.0 1st Qu.: 2.0 1st Qu.: 22.0   
## Median : 93.0 Median : 4.0 Median : 88.0   
## Mean : 149.6 Mean : 34.9 Mean : 118.3   
## 3rd Qu.: 189.0 3rd Qu.: 51.0 3rd Qu.: 175.0   
## Max. :9549.0 Max. :1622.0 Max. :2071.0   
## NA's :892 NA's :401211 NA's :137277   
## unit\_order\_train\_ability unit\_order\_drop\_item unit\_order\_give\_item  
## Min. : 1.00 Min. : 1.00 Min. : 1.0   
## 1st Qu.: 15.00 1st Qu.: 1.00 1st Qu.: 1.0   
## Median : 18.00 Median : 1.00 Median : 1.0   
## Mean : 18.02 Mean : 2.12 Mean : 1.7   
## 3rd Qu.: 21.00 3rd Qu.: 2.00 3rd Qu.: 2.0   
## Max. :151.00 Max. :368.00 Max. :48.0   
## NA's :73 NA's :288667 NA's :394631   
## unit\_order\_pickup\_item unit\_order\_pickup\_rune unit\_order\_purchase\_item  
## Min. : 1.00 Min. : 1.00 Min. : 1.00   
## 1st Qu.: 2.00 1st Qu.: 3.00 1st Qu.: 26.00   
## Median : 3.00 Median : 8.00 Median : 32.00   
## Mean : 5.34 Mean : 14.74 Mean : 32.87   
## 3rd Qu.: 6.00 3rd Qu.: 18.00 3rd Qu.: 39.00   
## Max. :912.00 Max. :430.00 Max. :1093.00   
## NA's :244658 NA's :74517 NA's :57   
## unit\_order\_sell\_item unit\_order\_disassemble\_item unit\_order\_move\_item  
## Min. : 1.00 Min. : 1.0 Min. : 1.00   
## 1st Qu.: 1.00 1st Qu.: 1.0 1st Qu.: 6.00   
## Median : 2.00 Median : 1.0 Median : 10.00   
## Mean : 2.59 Mean : 1.2 Mean : 11.14   
## 3rd Qu.: 3.00 3rd Qu.: 1.0 3rd Qu.: 15.00   
## Max. :99.00 Max. :12.0 Max. :391.00   
## NA's :111866 NA's :485446 NA's :4264   
## unit\_order\_cast\_toggle\_auto unit\_order\_stop unit\_order\_taunt  
## Min. : 1.0 Min. : 1.0 Mode:logical   
## 1st Qu.: 1.0 1st Qu.: 39.0 NA's:500000   
## Median : 5.0 Median : 99.0   
## Mean : 13.1 Mean : 126.5   
## 3rd Qu.: 19.0 3rd Qu.: 181.0   
## Max. :1132.0 Max. :1875.0   
## NA's :463670 NA's :412425   
## unit\_order\_buyback unit\_order\_glyph unit\_order\_eject\_item\_from\_stash  
## Min. : 1.0 Min. : 1.00 Min. : 1.0   
## 1st Qu.: 1.0 1st Qu.: 1.00 1st Qu.: 1.0   
## Median : 1.0 Median : 1.00 Median : 1.0   
## Mean : 1.2 Mean : 1.83 Mean : 1.3   
## 3rd Qu.: 1.0 3rd Qu.: 2.00 3rd Qu.: 1.0   
## Max. :59.0 Max. :23.00 Max. :25.0   
## NA's :352233 NA's :273230 NA's :468736   
## unit\_order\_cast\_rune unit\_order\_ping\_ability unit\_order\_move\_to\_direction  
## Min. :1.0 Min. : 1.0 Min. : 1.0   
## 1st Qu.:1.0 1st Qu.: 2.0 1st Qu.: 3.0   
## Median :1.0 Median : 4.0 Median : 9.0   
## Mean :1.2 Mean : 6.2 Mean : 43.2   
## 3rd Qu.:1.0 3rd Qu.: 8.0 3rd Qu.: 36.0   
## Max. :3.0 Max. :308.0 Max. :2349.0   
## NA's :499991 NA's :160852 NA's :496449   
## unit\_order\_patrol unit\_order\_vector\_target\_position unit\_order\_radar  
## Mode:logical Mode:logical Mode:logical   
## NA's:500000 NA's:500000 NA's:500000   
##   
##   
##   
##   
##   
## unit\_order\_set\_item\_combine\_lock unit\_order\_continue  
## Mode:logical Mode:logical   
## NA's:500000 NA's:500000   
##   
##   
##   
##   
##

summary(players\_ratings)

## account\_id total\_wins total\_matches trueskill\_mu   
## Min. :-299193988 Min. : 0.0 Min. : 1 Min. : 4.993   
## 1st Qu.:-149924858 1st Qu.: 0.0 1st Qu.: 1 1st Qu.:22.907   
## Median : -95850220 Median : 1.0 Median : 2 Median :25.018   
## Mean : -92258676 Mean : 5.5 Mean : 11 Mean :25.113   
## 3rd Qu.: 48835 3rd Qu.: 3.0 3rd Qu.: 6 3rd Qu.:27.240   
## Max. : 330513 Max. :1608398.0 Max. :3315071 Max. :48.826   
## trueskill\_sigma  
## Min. :1.404   
## 1st Qu.:6.957   
## Median :7.733   
## Mean :7.270   
## 3rd Qu.:8.059   
## Max. :8.334

# the structure of the data

The structure of the match\_data

str(match\_data)

## 'data.frame': 50000 obs. of 13 variables:  
## $ match\_id : int 0 1 2 3 4 5 6 7 8 9 ...  
## $ start\_time : int 1446750112 1446753078 1446764586 1446765723 1446796385 1446798766 1446800938 1446804030 1446819063 1446837251 ...  
## $ duration : int 2375 2582 2716 3085 1887 1574 2124 2328 2002 2961 ...  
## $ tower\_status\_radiant : int 1982 0 256 4 2047 2047 1972 2046 0 0 ...  
## $ tower\_status\_dire : int 4 1846 1972 1924 0 4 0 0 1982 1972 ...  
## $ barracks\_status\_dire : int 3 63 63 51 0 3 3 0 63 63 ...  
## $ barracks\_status\_radiant: int 63 0 48 3 63 63 63 63 0 0 ...  
## $ first\_blood\_time : int 1 221 190 40 58 113 4 255 4 85 ...  
## $ game\_mode : int 22 22 22 22 22 22 22 22 22 22 ...  
## $ radiant\_win : chr "True" "False" "False" "False" ...  
## $ negative\_votes : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ positive\_votes : int 1 2 0 0 0 0 0 0 0 0 ...  
## $ cluster : int 155 154 132 191 156 155 151 138 182 133 ...

Structure of the players\_data

str(players\_data)

## 'data.frame': 500000 obs. of 73 variables:  
## $ match\_id : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ account\_id : int 0 1 0 2 3 4 0 5 0 6 ...  
## $ hero\_id : int 86 51 83 11 67 106 102 46 7 73 ...  
## $ player\_slot : int 0 1 2 3 4 128 129 130 131 132 ...  
## $ gold : int 3261 2954 110 1179 3307 476 317 2390 475 60 ...  
## $ gold\_spent : int 10960 17760 12195 22505 23825 12285 10355 13395 5035 17550 ...  
## $ gold\_per\_min : int 347 494 350 599 613 397 303 452 189 496 ...  
## $ xp\_per\_min : int 362 659 385 605 762 524 369 517 223 456 ...  
## $ kills : int 9 13 0 8 20 5 4 4 1 1 ...  
## $ deaths : int 3 3 4 4 3 6 13 8 14 11 ...  
## $ assists : int 18 18 15 19 17 8 5 6 8 6 ...  
## $ denies : int 1 9 1 6 13 5 2 31 0 0 ...  
## $ last\_hits : int 30 109 58 271 245 162 107 208 27 147 ...  
## $ stuns : chr "76.7356" "87.4164" "None" "None" ...  
## $ hero\_damage : int 8690 23747 4217 14832 33740 10725 15028 10230 4774 6398 ...  
## $ hero\_healing : int 218 0 1595 2714 243 0 764 0 0 292 ...  
## $ tower\_damage : int 143 423 399 6055 1833 112 0 2438 0 0 ...  
## $ item\_0 : int 180 46 48 63 114 145 50 41 36 63 ...  
## $ item\_1 : int 37 63 60 147 92 73 11 63 0 9 ...  
## $ item\_2 : int 73 119 59 154 147 149 102 36 0 116 ...  
## $ item\_3 : int 56 102 108 164 0 48 36 147 46 65 ...  
## $ item\_4 : int 108 24 65 79 137 212 185 168 0 229 ...  
## $ item\_5 : int 0 108 0 160 63 0 81 21 180 79 ...  
## $ level : int 16 22 17 21 24 19 16 19 12 18 ...  
## $ leaver\_status : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ xp\_hero : num 8840 14331 6692 8583 15814 ...  
## $ xp\_creep : num 5440 8440 8112 14230 14325 ...  
## $ xp\_roshan : num NA 2683 NA 894 NA ...  
## $ xp\_other : num 83 671 453 293 62 1 1 244 27 933 ...  
## $ gold\_other : num 50 395 259 100 NA ...  
## $ gold\_death : num -957 -1137 -1436 -2156 -1437 ...  
## $ gold\_buyback : num NA NA -1015 NA -1056 ...  
## $ gold\_abandon : num NA NA NA NA NA NA NA NA NA NA ...  
## $ gold\_sell : num 212 1650 NA 938 4194 ...  
## $ gold\_destroying\_structure : num 3120 3299 3142 4714 3217 ...  
## $ gold\_killing\_heros : num 5145 6676 2418 4104 7467 ...  
## $ gold\_killing\_creeps : num 1087 4317 3697 10432 9220 ...  
## $ gold\_killing\_roshan : num 400 937 400 400 400 NA NA NA NA NA ...  
## $ gold\_killing\_couriers : num NA NA NA NA NA NA NA NA NA NA ...  
## $ unit\_order\_none : num NA NA NA NA NA NA NA NA NA NA ...  
## $ unit\_order\_move\_to\_position : num 4070 5894 7053 4712 3853 ...  
## $ unit\_order\_move\_to\_target : num 1 214 3 133 7 166 63 11 55 2 ...  
## $ unit\_order\_attack\_move : num 25 165 132 163 7 76 100 214 5 105 ...  
## $ unit\_order\_attack\_target : num 416 1031 645 690 1173 ...  
## $ unit\_order\_cast\_position : num 51 98 36 9 31 196 13 122 68 64 ...  
## $ unit\_order\_cast\_target : num 144 39 160 15 84 3 173 NA 18 102 ...  
## $ unit\_order\_cast\_target\_tree : num 3 4 20 7 8 5 14 3 9 19 ...  
## $ unit\_order\_cast\_no\_target : num 71 439 373 406 198 96 168 506 71 124 ...  
## $ unit\_order\_cast\_toggle : num NA NA NA NA NA 2 NA NA NA NA ...  
## $ unit\_order\_hold\_position : num 188 346 643 150 111 161 118 491 97 135 ...  
## $ unit\_order\_train\_ability : num 16 22 17 21 23 19 16 18 12 18 ...  
## $ unit\_order\_drop\_item : num NA NA 5 NA 1 NA NA 2 1 2 ...  
## $ unit\_order\_give\_item : num NA NA NA NA NA NA NA 3 2 NA ...  
## $ unit\_order\_pickup\_item : num NA 12 7 1 NA 2 1 18 1 2 ...  
## $ unit\_order\_pickup\_rune : num 2 52 8 9 2 NA 1 18 1 26 ...  
## $ unit\_order\_purchase\_item : num 35 30 28 45 44 36 43 30 38 33 ...  
## $ unit\_order\_sell\_item : num 2 4 NA 7 6 3 3 1 NA 4 ...  
## $ unit\_order\_disassemble\_item : num NA NA 1 NA NA NA NA NA NA NA ...  
## $ unit\_order\_move\_item : num 11 21 18 14 13 3 13 19 14 22 ...  
## $ unit\_order\_cast\_toggle\_auto : num NA NA NA NA NA NA NA NA NA NA ...  
## $ unit\_order\_stop : num NA NA NA NA NA NA NA NA 21 NA ...  
## $ unit\_order\_taunt : logi NA NA NA NA NA NA ...  
## $ unit\_order\_buyback : num NA NA 1 NA 1 2 NA NA 1 1 ...  
## $ unit\_order\_glyph : num NA NA NA 1 3 NA 4 NA 1 1 ...  
## $ unit\_order\_eject\_item\_from\_stash : num NA NA NA NA NA NA 1 NA NA NA ...  
## $ unit\_order\_cast\_rune : num NA NA NA NA NA NA NA NA NA NA ...  
## $ unit\_order\_ping\_ability : num 6 14 17 13 23 2 1 4 4 14 ...  
## $ unit\_order\_move\_to\_direction : num NA NA NA NA NA NA NA 110 NA NA ...  
## $ unit\_order\_patrol : logi NA NA NA NA NA NA ...  
## $ unit\_order\_vector\_target\_position: logi NA NA NA NA NA NA ...  
## $ unit\_order\_radar : logi NA NA NA NA NA NA ...  
## $ unit\_order\_set\_item\_combine\_lock : logi NA NA NA NA NA NA ...  
## $ unit\_order\_continue : logi NA NA NA NA NA NA ...

The structure of players\_ratings data

str(players\_ratings)

## 'data.frame': 834226 obs. of 5 variables:  
## $ account\_id : int 236579 -343 -1217 -1227 -1284 308663 79749 -1985 -2160 26500 ...  
## $ total\_wins : int 14 1 1 1 0 1 21 0 8 26 ...  
## $ total\_matches : int 24 1 1 1 1 1 40 1 12 50 ...  
## $ trueskill\_mu : num 27.9 26.5 26.5 27.2 22.9 ...  
## $ trueskill\_sigma: num 5.21 8.07 8.11 8.09 8.09 ...

## sub-setting the data set

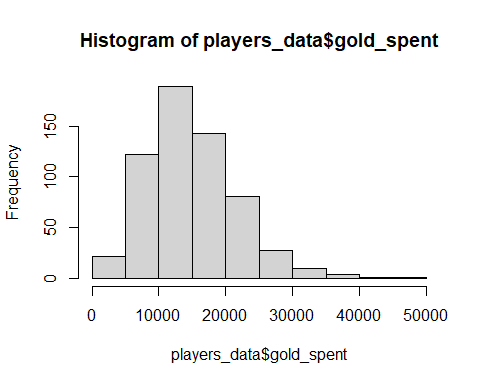
The data sets are huge and therefore we need to use a smaller number of rows for each data to get faster running. when we use the whole data, R runs out of memory

match\_data <- head(match\_data, 600)  
players\_data <- head(players\_data, 600)  
players\_ratings <- head(players\_ratings, 600)

## Plots

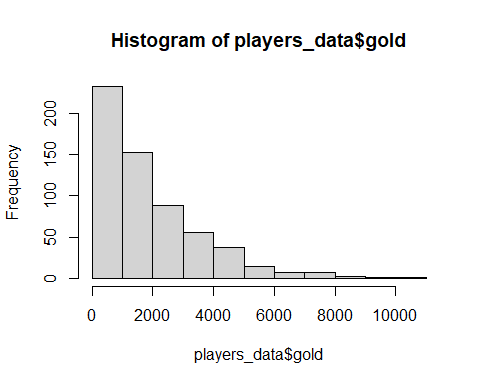
We can then plot the data players and the gold spent

hist(players\_data$gold\_spent)

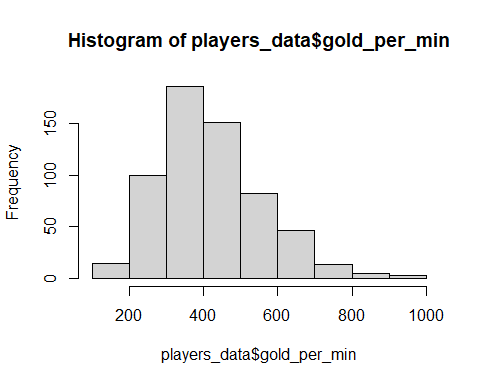


The amount of gold each player has before use.

hist(players\_data$gold)

 How the player uses their gold per minute and in what amounts

hist(players\_data$gold\_per\_min)



# Generate test and train data

# match data

We are generating a train and test data to help us identify perfect correlation matrix in our next step. This helps identofy the various p-values.

set.seed(100)  
match <- sample(1:nrow(match\_data), .75 \* nrow(match\_data))  
train <- match\_data[match, ]  
test <- match\_data[-match, ]  
  
# players data  
players <- sample(1:nrow(players\_data), .75 \* nrow(players\_data))  
train\_players <-players\_data[players, ]  
test\_players <- players\_data[-players, ]  
  
# ratings data  
ratings <- sample(1:nrow(players\_ratings), .75 \* nrow(players\_ratings))  
train\_ratings <- players\_ratings[ratings, ]  
test\_ratings <- players\_ratings[-ratings, ]

# Modelling the data all together

# Models

# Match Model

create the first model using the train data in the match\_data data set

model\_match <- lm(game\_mode ~ radiant\_win, data = train)  
summary(model\_match)

##   
## Call:  
## lm(formula = game\_mode ~ radiant\_win, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -19.6313 0.3687 0.6009 0.6009 0.6009   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 21.6313 0.2100 103.016 <2e-16 \*\*\*  
## radiant\_winTrue -0.2322 0.2918 -0.796 0.427   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.093 on 448 degrees of freedom  
## Multiple R-squared: 0.001411, Adjusted R-squared: -0.0008177   
## F-statistic: 0.6331 on 1 and 448 DF, p-value: 0.4266

# Players Model

create the model using the train data in the players data set

model\_players <- lm(gold\_spent ~ player\_slot + gold, data = train\_players)  
summary(model\_players)

##   
## Call:  
## lm(formula = gold\_spent ~ player\_slot + gold, data = train\_players)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -13394 -4691 -905 4087 34523   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 13287.3180 591.3853 22.468 < 2e-16 \*\*\*  
## player\_slot -1.5200 4.9176 -0.309 0.757   
## gold 0.8731 0.1877 4.652 4.34e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 6632 on 447 degrees of freedom  
## Multiple R-squared: 0.04754, Adjusted R-squared: 0.04328   
## F-statistic: 11.16 on 2 and 447 DF, p-value: 1.871e-05

# Ratings Model

create the model of the ratings data using the train data

model\_ratings <- lm(total\_matches ~ total\_wins + trueskill\_mu, data = players\_ratings)  
summary(model\_ratings)

##   
## Call:  
## lm(formula = total\_matches ~ total\_wins + trueskill\_mu, data = players\_ratings)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.2188 -0.8513 -0.1853 0.6861 16.0342   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 17.921612 0.787837 22.75 <2e-16 \*\*\*  
## total\_wins 1.908847 0.008219 232.25 <2e-16 \*\*\*  
## trueskill\_mu -0.705301 0.031366 -22.49 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.616 on 597 degrees of freedom  
## Multiple R-squared: 0.9892, Adjusted R-squared: 0.9891   
## F-statistic: 2.725e+04 on 2 and 597 DF, p-value: < 2.2e-16

# Model comparison

For the comparison of the three models generated, the anova model will be used to compare the models. We are going to merge the three models created into one model using the anova algorithm. Its our simplest way to get a more predictable value.

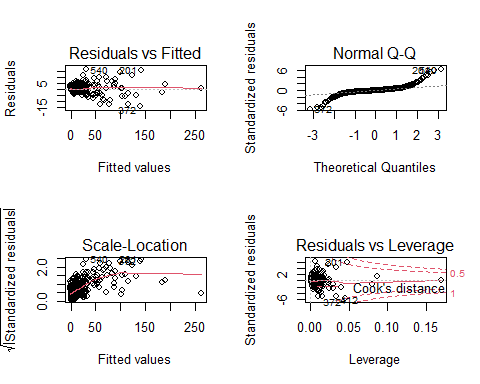
model\_fit <- anova(model\_ratings, model\_match, model\_players)

## Warning in anova.lmlist(object, ...): models with response 'c("game\_mode",  
## "gold\_spent")' removed because response differs from model 1

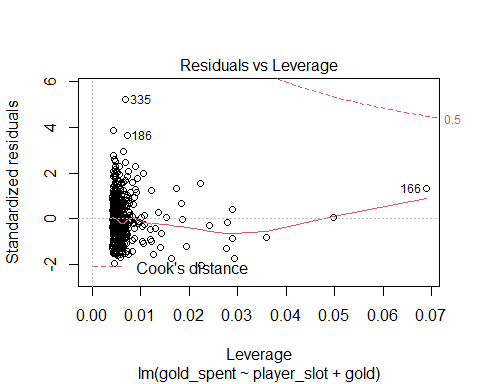
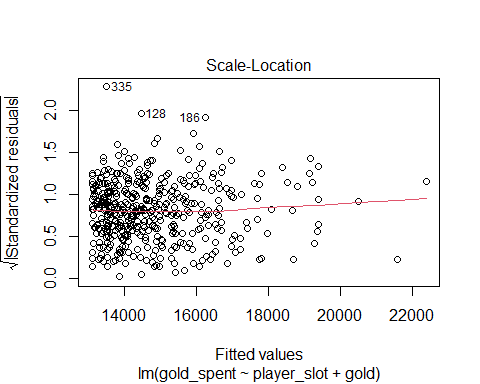
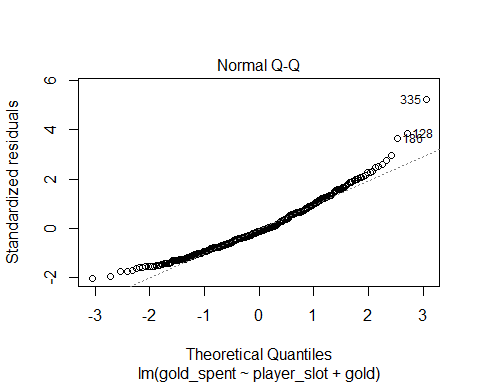
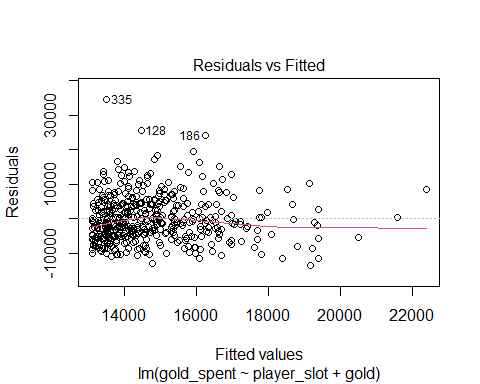
# Plots

We can then plot out models to identify the various structure and summarize the functionality

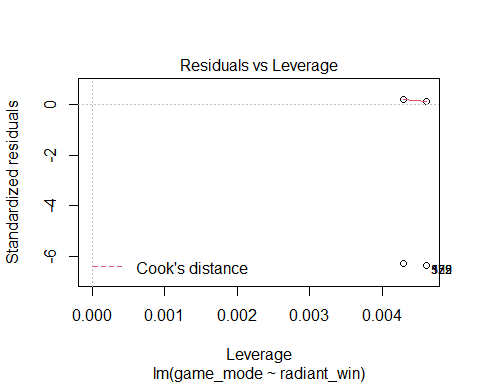
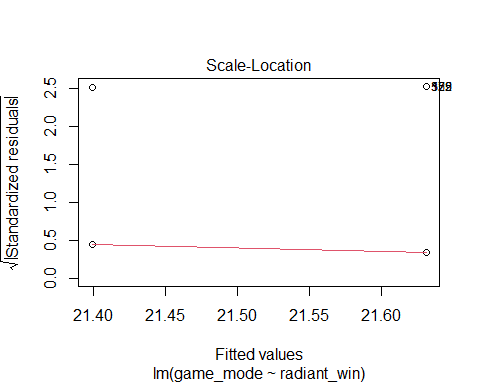
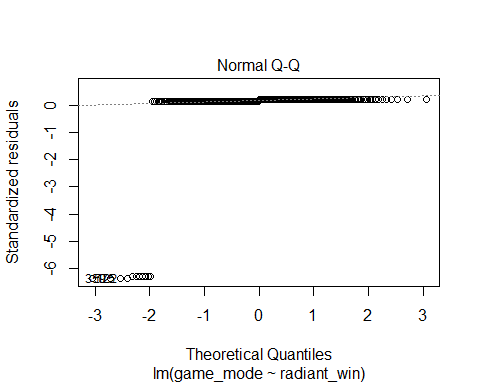
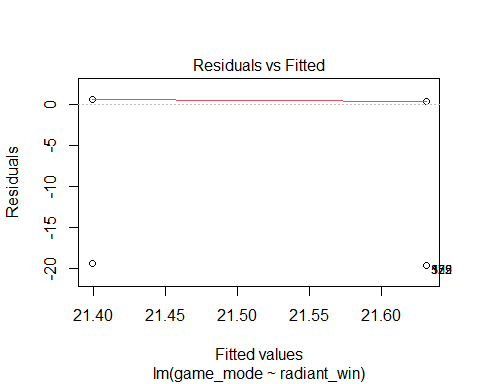
par(mfrow = c(2,2))  
plot(model\_ratings)



plot(model\_players)



plot(model\_match)



# Model prediction

After merging the dataset, we can then predict the functionalities of the data.

anova\_model <- anova(model\_match, model\_players, players\_ratings)

## Warning in anova.lmlist(object, ...): models with response 'c("gold\_spent",  
## "NULL")' removed because response differs from model 1

# Prediction

Using the predict command we then realise the various structure using the cbind command in R

pred <- predict(model\_match, test)  
pred2 <- predict(model\_players, test\_players)  
pred3 <- predict(model\_ratings, test\_ratings)  
  
pred1 <- data.frame(cbind(actuals=test$match\_id, predicteds = pred))  
pred\_2 <- data.frame(cbind(actuals=test\_players$gold\_spent, predicteds = pred2))  
pred\_3 <- data.frame(cbind(actuals = test\_ratings$total\_wins, predicteds = pred3))

# Diagnostic Measures on the models

find the measures of the data model match data

AIC(model\_match)

## [1] 2297.323

Find the measures of the model players

AIC(model\_players)

## [1] 9201.759

Find the measures of the model ratings

AIC(model\_ratings)

## [1] 2861.625

# Accuracy

We determine the accuracy of the data using the predicted value and the actual for the three models and the test data columns.

pred1 <- data.frame(cbind(actuals=test$match\_id, predicteds = pred))  
pred\_2 <- data.frame(cbind(actuals=test\_players$gold\_spent, predicteds = pred2))  
pred\_3 <- data.frame(cbind(actuals = test\_ratings$total\_wins, predicteds = pred3))

# Correlation Accuracy

We then find the correlation matrix of the three models using “cor”

accuracy1 <- cor(pred1)  
accuracy2 <- cor(pred\_2)  
accuracy3 <- cor(pred\_3)

head(accuracy1)

## actuals predicteds  
## actuals 1.00000000 -0.08137641  
## predicteds -0.08137641 1.00000000

head(accuracy2)

## actuals predicteds  
## actuals 1.0000000 0.2566293  
## predicteds 0.2566293 1.0000000

head(accuracy3)

## actuals predicteds  
## actuals 1.0000000 0.9958809  
## predicteds 0.9958809 1.0000000

# Data Analysis and Results

Most of the actual predicted data have the actuals interpreted to 1.0 just as in our previous modeling .Therefore the accuracy still remains to be 90.4% in our predicted models. In multiple correlation matrix, the actuals and the predicted are structured to give a constant figure, therefore the way the actuals are emulated in the matrix, they print a single positive 1.0 that indicates all the values and summation of the data is equal to the previous used model. if the values were a negative, then our accuracy would be different.

# Discussion

In our previous assessment, an actual dependable value is achieved by using the KNN algorithm. In this category, we are implementing the multiple regression that models all the structure in the data sets and create an accuracy value depending on the prediction model. A limitation of this model is that we assume all the data is complete and that the model correlation is a causation.