LoL-Statistics

Andrew Roddy

2025-11-24

MatchStatsTbl.csv

Import code and print off column and row count

```
match_stats <- read.csv("data/MatchStatsTbl.csv")
row_count <- nrow(match_stats)
col_count <- ncol(match_stats)
cat("There are", row_count, "rows.\n")
```

```
## There are 150505 rows.
```

```
cat("There are", col_count, "columns.\n")
```

```
## There are 31 columns.
```

```
team_stats <- read.csv("data/TeamMatchTbl.csv")
row_count2 <- nrow(team_stats)
col_count2 <- ncol(team_stats)
cat("There are", row_count2, "rows.\n")
```

```
## There are 68676 rows.
```

```
cat("There are", col_count2, "columns.\n")
```

```
## There are 24 columns.
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
summoner_match <- read.csv("data/other/SummonerMatchTbl.csv")
match_stats <- match_stats %>%
  left_join(summoner_match, by = c("SummonerMatchFk" = "SummonerMatchId"))

champions <- read.csv("data/keys/ChampionTbl.csv")
match_stats <- match_stats %>%
  left_join(champions, by = c("ChampionFk" = "ChampionId"))
```

```r
# Removes unimportant columns of data
match_stats[, c(
  "item1", "item2", "item3", "item4", "item5", "item6",
  "PrimarySlot1", "PrimarySlot2", "PrimarySlot3",
  "SecondarySlot1", "SecondarySlot2",
  "SummonerSpell1", "SummonerSpell2"
)] <- list(NULL)

names <- colnames(match_stats)
print(names)
```

```
##  [1] "MatchStatsId"          "SummonerMatchFk"      "MinionsKilled"
##  [4] "DmgDealt"              "DmgTaken"             "TurretDmgDealt"
##  [7] "TotalGold"             "Lane"                 "Win"
## [10] "kills"                 "deaths"               "assists"
## [13] "PrimaryKeyStone"       "CurrentMasteryPoints" "EnemyChampionFk"
## [16] "DragonKills"           "BaronKills"           "visionScore"
## [19] "SummonerFk"            "MatchFk"              "ChampionFk"
## [22] "ChampionName"
```

Multiple regression to test if

```r
library(ggplot2)
library(scales)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

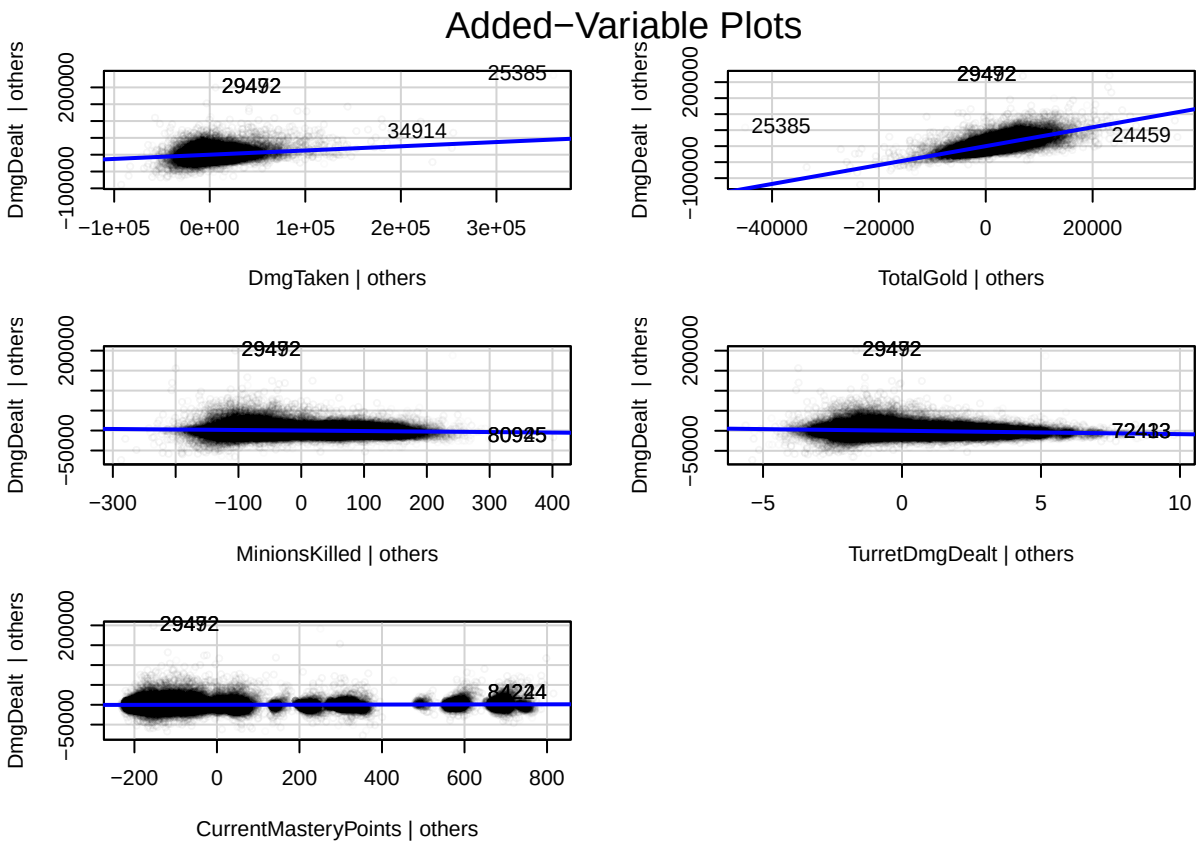
```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
# make simple regression model
multiple_regression <- lm(DmgDealt ~
    DmgTaken + TotalGold + MinionsKilled + TurretDmgDealt +
      CurrentMasteryPoints,
  data = match_stats
)
# Print off if this matters or not
summary(multiple_regression)
```

```
##
## Call:
## lm(formula = DmgDealt ~ DmgTaken + TotalGold + MinionsKilled +
##     TurretDmgDealt + CurrentMasteryPoints, data = match_stats)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -77533  -5554   -689   4391 198809
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.201e+04  7.706e+01 -155.89   <2e-16 ***
## DmgTaken              1.252e-01  1.829e-03   68.47   <2e-16 ***
## TotalGold             2.954e+00  7.584e-03  389.47   <2e-16 ***
## MinionsKilled        -1.285e+01  3.176e-01  -40.45   <2e-16 ***
## TurretDmgDealt       -8.396e+02  1.473e+01  -57.02   <2e-16 ***
## CurrentMasteryPoints  1.366e+00  1.020e-01   13.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9923 on 150499 degrees of freedom
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6846
## F-statistic: 6.535e+04 on 5 and 150499 DF,  p-value: < 2.2e-16
```

```r
# Creates the variable plots
 #000000 point color
 #0.01 : 1 percent opacity

# Commented out for now
 avPlots(multiple_regression, col = alpha("#000000", 0.02))
```




```
#KEYSTONE KEY
#8465 - Guardian
#8214 - SummonAery
#8112 - Electrocute
#0 - MissingData?
#8128 - Dark Harvest
#8010 - Conquerer
#8229 - ArcaneComet
#8369 - FirstStrike
#8008 - LethalTempo
#8005 - PressTheAttack
#8439 - Afterschock
#8230 - PhaseRush
#9923 - HailOfBlades
#8021 - FleetFootwork
#8351 - GlacialAugment
#8437 - GraspOfTheUndying
#8360 - UnsealedSpellbook
#Remove outliers
dmg_dealt <- match_stats$DmgDealt
q1 <- quantile(dmg_dealt, 0.25)
q3 <- quantile(dmg_dealt, 0.75)
iqr <- q3 - q1

lower_bound <- q1 - 1.5 * iqr
upper_bound <- q3 + 1.5 * iqr
```

```r
filtered <- match_stats[
  dmg_dealt >= lower_bound & dmg_dealt <= upper_bound &
    match_stats$PrimaryKeyStone != "0" & match_stats$Lane != "NONE",
]

# 2 Way ANOVA on Keystone and Role
filtered$PrimaryKeyStone <- factor(filtered$PrimaryKeyStone)
filtered$Lane <- factor(filtered$Lane)
model <- aov(DmgDealt ~ PrimaryKeyStone * Lane, data = filtered)
summary(model)
```

```
##                          Df    Sum Sq   Mean Sq F value Pr(>F)
## PrimaryKeyStone          15 2.299e+12 1.533e+11 1287.77 <2e-16 ***
## Lane                      4 5.753e+11 1.438e+11 1208.36 <2e-16 ***
## PrimaryKeyStone:Lane     60 3.482e+11 5.804e+09   48.76 <2e-16 ***
## Residuals            107096 1.275e+13 1.190e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Results show a high F value meaning that KeyStone has a large
# effect on average damage even with the large amount of noise
# (random player variation still makes most of the difference)

# P-Values all very low so all very signifigant
# Residuals are very high, meaning that KeyStone and
# Lane are primarily not the biggest effects and
# most is left up to other factors
# (Likely player randomness and maybe gold earned)

#Find Individual stats
key_stone_stats <- TukeyHSD(model, "PrimaryKeyStone")
key_stone_stats
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = DmgDealt ~ PrimaryKeyStone * Lane, data = filtered)
##
## $PrimaryKeyStone
##                   diff         lwr          upr     p adj
## 8008-8005   -750.6240  -1274.9536   -226.29444 0.0001076
## 8010-8005  -1767.4492  -2272.1368  -1262.76164 0.0000000
## 8021-8005  -1067.1114  -1745.9271   -388.29557 0.0000085
## 8112-8005   -938.1620  -1505.9070   -370.41687 0.0000018
## 8128-8005   4002.3982   3369.6884   4635.10804 0.0000000
## 8214-8005 -12517.5384 -13134.9457 -11900.13110 0.0000000
## 8229-8005   -621.3516  -1193.3153    -49.38781 0.0182899
## 8230-8005  -2279.9575  -2991.5834  -1568.33167 0.0000000
## 8351-8005 -15708.6799 -16894.9248 -14522.43505 0.0000000
## 8360-8005 -15190.1102 -16462.7747 -13917.44566 0.0000000
## 8369-8005    998.2429    239.5255   1756.96031 0.0007246
## 8437-8005  -1188.7295  -1801.8765   -575.58247 0.0000000
```

```
## 8439-8005 -12125.1389 -12821.6865 -11428.59140 0.0000000
## 8465-8005 -16449.1140 -17511.6807 -15386.54737 0.0000000
## 9923-8005  -4660.1341  -5489.0140  -3831.25423 0.0000000
## 8010-8008  -1016.8252  -1419.5034   -614.14700 0.0000000
## 8021-8008   -316.4873   -923.3078    290.33312 0.9214820
## 8112-8008   -187.5379   -666.8861    291.81023 0.9944283
## 8128-8008   4753.0223   4198.2594   5307.78516 0.0000000
## 8214-8008 -11766.9144 -12304.1592 -11229.66959 0.0000000
## 8229-8008    129.2725   -355.0649    613.60985 0.9999378
## 8230-8008  -1529.3335  -2172.6465   -886.02056 0.0000000
## 8351-8008 -14958.0559 -16104.6234 -13811.48838 0.0000000
## 8360-8008 -14439.4862 -15675.2512 -13203.72114 0.0000000
## 8369-8008   1748.8670   1053.8186   2443.91533 0.0000000
## 8437-8008   -438.1054   -970.4488     94.23789 0.2575539
## 8439-8008 -11374.5149 -12001.1078 -10747.92202 0.0000000
## 8465-8008 -15698.4900 -16716.5706 -14680.40941 0.0000000
## 9923-8008  -3909.5101  -4680.5367  -3138.48347 0.0000000
## 8021-8010    700.3379    110.4065   1290.26926 0.0048637
## 8112-8010    829.2873    371.5071   1287.06741 0.0000001
## 8128-8010   5769.8474   5233.6107   6306.08423 0.0000000
## 8214-8010 -10750.0892 -11268.1819 -10231.99656 0.0000000
## 8229-8010   1146.0977    683.0958   1609.09950 0.0000000
## 8230-8010   -512.5083  -1139.9153    114.89869 0.2695118
## 8351-8010 -13941.2307 -15078.9499 -12803.51147 0.0000000
## 8360-8010 -13422.6610 -14650.2209 -12195.10112 0.0000000
## 8369-8010   2765.6921   2085.3391   3446.04518 0.0000000
## 8437-8010    578.7197     65.7115   1091.72796 0.0107697
## 8439-8010 -10357.6897 -10967.9410  -9747.43841 0.0000000
## 8465-8010 -14681.6648 -15689.7701 -13673.55962 0.0000000
## 9923-8010  -2892.6849  -3650.4910  -2134.87882 0.0000000
## 8112-8021    128.9494   -515.7552    773.65398 0.9999987
## 8128-8021   5069.5096   4366.9207   5772.09852 0.0000000
## 8214-8021 -11450.4271 -12139.2676 -10761.58651 0.0000000
## 8229-8021    445.7598   -202.6629   1094.18253 0.5818028
## 8230-8021  -1212.8462  -1987.2627   -438.42970 0.0000095
## 8351-8021 -14641.5686 -15866.5120 -13416.62514 0.0000000
## 8360-8021 -14122.9988 -15431.8092 -12814.18849 0.0000000
## 8369-8021   2065.3543   1247.4536   2883.25499 0.0000000
## 8437-8021   -121.6181   -806.6428    563.40652 0.9999997
## 8439-8021 -11058.0276 -11818.6115 -10297.44360 0.0000000
## 8465-8021 -15382.0027 -16487.6054 -14276.39994 0.0000000
## 9923-8021  -3593.0228  -4476.3978  -2709.64777 0.0000000
## 8128-8112   4940.5602   4344.5947   5536.52564 0.0000000
## 8214-8112 -11579.3765 -12159.0703 -10999.68264 0.0000000
## 8229-8112    316.8104   -214.2222    847.84302 0.8003848
## 8230-8112  -1341.7956  -2020.9601   -662.63102 0.0000000
## 8351-8112 -14770.5180 -15937.5784 -13603.45755 0.0000000
## 8360-8112 -14251.9482 -15506.7503 -12997.14621 0.0000000
## 8369-8112   1936.4049   1208.0470   2664.76275 0.0000000
## 8437-8112   -250.5675   -825.7217    324.58670 0.9835688
## 8439-8112 -11186.9770 -11850.3257 -10523.62821 0.0000000
## 8465-8112 -15510.9521 -16552.0578 -14469.84640 0.0000000
## 9923-8112  -3721.9722  -4523.1557  -2920.78867 0.0000000
## 8214-8128 -16519.9367 -17163.3899 -15876.48337 0.0000000
```

```
## 8229-8128  -4623.7498  -5223.7355  -4023.76408 0.0000000
## 8230-8128  -6282.3558  -7016.6934  -5548.01818 0.0000000
## 8351-8128 -19711.0782 -20911.0853 -18511.07098 0.0000000
## 8360-8128 -19192.5084 -20478.0104 -17907.00645 0.0000000
## 8369-8128  -3004.1553  -3784.2146  -2224.09608 0.0000000
## 8437-8128  -5191.1277  -5830.4943  -4551.76116 0.0000000
## 8439-8128 -16127.5372 -16847.2724 -15407.80194 0.0000000
## 8465-8128 -20451.5123 -21529.4215 -19373.60307 0.0000000
## 9923-8128  -8662.5324  -9510.9911  -7814.07363 0.0000000
## 8229-8214  11896.1869  11312.3607  12480.01300 0.0000000
## 8230-8214  10237.5809   9516.3862  10958.77559 0.0000000
## 8351-8214  -3191.1415  -4383.1513  -1999.13172 0.0000000
## 8360-8214  -2672.5718  -3950.6115  -1394.53208 0.0000000
## 8369-8214  13515.7813  12748.0818  14283.48083 0.0000000
## 8437-8214  11328.8089  10704.5817  11953.03622 0.0000000
## 8439-8214    392.3995   -313.9212   1098.72017 0.8749503
## 8465-8214  -3931.5756  -5000.5744  -2862.57686 0.0000000
## 9923-8214   7857.4043   7020.2948   8694.51380 0.0000000
## 8230-8229  -1658.6060  -2341.3010   -975.91093 0.0000000
## 8351-8229 -15087.3284 -16256.4468 -13918.20988 0.0000000
## 8360-8229 -14568.7586 -15825.4751 -13312.04221 0.0000000
## 8369-8229   1619.5945    887.9434   2351.24551 0.0000000
## 8437-8229   -567.3779  -1146.6968     11.94100 0.0623801
## 8439-8229 -11503.7874 -12170.7504 -10836.82439 0.0000000
## 8465-8229 -15827.7625 -16871.1747 -14784.35025 0.0000000
## 9923-8229  -4038.7826  -4842.9611  -3234.60408 0.0000000
## 8351-8230 -13428.7224 -14672.1478 -12185.29691 0.0000000
## 8360-8230 -12910.1527 -14236.2767 -11584.02859 0.0000000
## 8369-8230   3278.2005   2432.8709   4123.53004 0.0000000
## 8437-8230   1091.2281    373.6772   1808.77895 0.0000221
## 8439-8230  -9845.1814 -10635.1868  -9055.17602 0.0000000
## 8465-8230 -14169.1565 -15295.2018 -13043.11123 0.0000000
## 9923-8230  -2380.1766  -3289.0065  -1471.34662 0.0000000
## 8360-8351    518.5697  -1112.1879   2149.32737 0.9994584
## 8369-8351  16706.9228  15435.9596  17977.88609 0.0000000
## 8437-8351  14519.9504  13330.1417  15709.75914 0.0000000
## 8439-8351   3583.5410   2348.6831   4818.39886 0.0000000
## 8465-8351   -740.4341  -2213.0996    732.23129 0.9413004
## 9923-8351  11048.5458   9734.4920  12362.59959 0.0000000
## 8369-8360  16188.3531  14836.3747  17540.33158 0.0000000
## 8437-8360  14001.3807  12725.3937  15277.36776 0.0000000
## 8439-8360   3064.9713   1746.8771   4383.06540 0.0000000
## 8465-8360  -1259.0038  -2802.1311    284.12337 0.2714797
## 9923-8360  10529.9761   9137.4117  11922.54046 0.0000000
## 8437-8369  -2186.9724  -2951.2498  -1422.69498 0.0000000
## 8439-8369 -13123.3818 -13956.0578 -12290.70593 0.0000000
## 8465-8369 -17447.3570 -18603.7387 -16290.97526 0.0000000
## 9923-8369  -5658.3770  -6604.5338  -4712.22029 0.0000000
## 8439-8437 -10936.4094 -11639.0092 -10233.80973 0.0000000
## 8465-8437 -15260.3846 -16326.9284 -14193.84069 0.0000000
## 9923-8437  -3471.4046  -4305.3769  -2637.43236 0.0000000
## 8465-8439  -4323.9751  -5440.5525  -3207.39774 0.0000000
## 9923-8439   7465.0048   6567.9324   8362.07726 0.0000000
## 9923-8465  11788.9799  10585.3984  12992.56144 0.0000000
```

```r
lane_stats <- TukeyHSD(model, "Lane")
lane_stats
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = DmgDealt ~ PrimaryKeyStone * Lane, data = filtered)
##
## $Lane
##                       diff         lwr        upr     p adj
## JUNGLE-BOTTOM  -1412.1850 -1677.22440 -1147.1456 0.0000000
## MIDDLE-BOTTOM   3012.5987  2753.75008  3271.4473 0.0000000
## SUPPORT-BOTTOM -2915.6648 -3298.85921 -2532.4704 0.0000000
## TOP-BOTTOM      3272.9074  3011.93488  3533.8799 0.0000000
## MIDDLE-JUNGLE   4424.7836  4150.19879  4699.3685 0.0000000
## SUPPORT-JUNGLE -1503.4798 -1897.47501 -1109.4846 0.0000000
## TOP-JUNGLE      4685.0924  4408.50440  4961.6803 0.0000000
## SUPPORT-MIDDLE -5928.2635 -6318.12101 -5538.4059 0.0000000
## TOP-MIDDLE       260.3087   -10.35268   530.9701 0.0661567
## TOP-SUPPORT     6188.5722  5797.30123  6579.8432 0.0000000
```

```r
#NEED TO DO: GRAPH AND VISULIZE DATA PROPERLY
```

```r
#Logistic Regression on a bunch of team Data
blue_barons <- team_stats$BlueBaronKills
blue_dragons <- team_stats$BlueDragonKills
blue_heralds <- team_stats$BlueRiftHeraldKills
blue_kills <- team_stats$BlueKills

model <- glm(
  BlueWin ~ blue_barons + blue_dragons + blue_heralds + blue_kills,
  data = team_stats,
  family = binomial
)

summary(model)
```

```
##
## Call:
## glm(formula = BlueWin ~ blue_barons + blue_dragons + blue_heralds +
##     blue_kills, family = binomial, data = team_stats)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.8943909  0.0288116 -100.46   <2e-16 ***
## blue_barons   0.3734663  0.0225364   16.57   <2e-16 ***
## blue_dragons  0.3968396  0.0087527   45.34   <2e-16 ***
## blue_heralds  0.3410325  0.0210757   16.18   <2e-16 ***
## blue_kills    0.0694563  0.0006547  106.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 93662  on 68675  degrees of freedom
## Residual deviance: 71484  on 68671  degrees of freedom
## AIC: 71494
##
## Number of Fisher Scoring iterations: 4
```

```
# Results give us a legit function,
# need to convert to individual winrate multipliers
odd_mults <- exp(coef(model)[-1])
odd_mults
```

```
##  blue_barons blue_dragons blue_heralds   blue_kills
##     1.452762     1.487117     1.406399     1.071925
```

```
#Probability Conversion = Mults/1+mult Can build predictor model off of this
winrate <- 0.5 #50% chance of winning a game
barons <- -2
dragons <- -1
heralds <- 1
kills <- 5
odds <- winrate / (1 - winrate) *
  odd_mults["blue_barons"]^barons *
  odd_mults["blue_dragons"]^dragons *
  odd_mults["blue_heralds"]^heralds *
  odd_mults["blue_kills"]^kills

probability <- odds / (1 + odds)
probability
```

```
## blue_barons
##   0.3880638
```