

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
```

```

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"))

# 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

```

library(ggplot2)
library(scales)
library(car)

## Loading required package: carData

## 
## Attaching package: 'car'

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

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

```

```

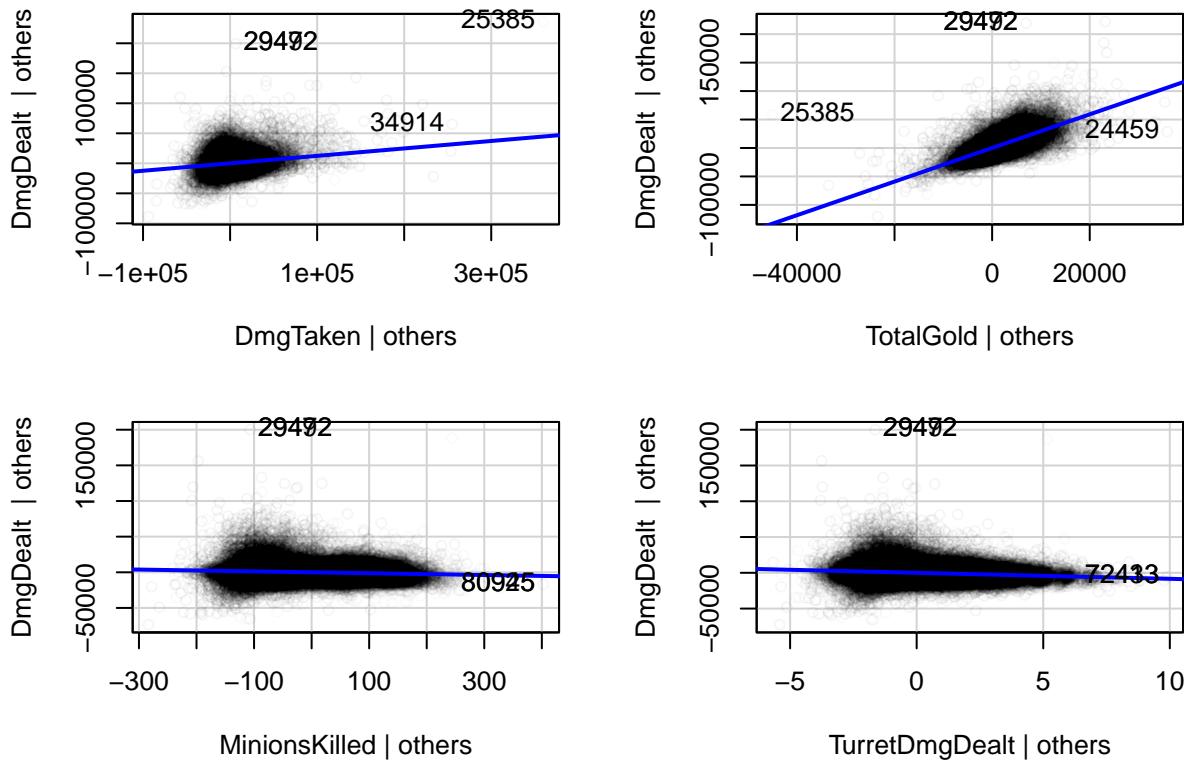
## Call:
## lm(formula = DmgDealt ~ DmgTaken + TotalGold + MinionsKilled +
##      TurretDmgDealt, data = match_stats)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -77051 -5559   -702   4379 198600 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.171e+04 7.372e+01 -158.85 <2e-16 ***
## DmgTaken     1.241e-01 1.828e-03   67.88 <2e-16 ***
## TotalGold    2.954e+00 7.589e-03   389.27 <2e-16 ***
## MinionsKilled -1.272e+01 3.176e-01  -40.03 <2e-16 ***
## TurretDmgDealt -8.420e+02 1.473e+01  -57.15 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9929 on 150500 degrees of freedom
## Multiple R-squared:  0.6843, Adjusted R-squared:  0.6843 
## F-statistic: 8.154e+04 on 4 and 150500 DF,  p-value: < 2.2e-16

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

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

```

Added-Variable Plots



```

library(ggplot2)
library(dplyr)

#Definition list for each rune name and ID
rune_stones_lookup <- data.frame(
  rune = c(
    "Guardian", "SummonAery", "Electrocute", "DarkHarvest", "Conquerer",
    "ArcaneComet", "FirstStrike", "LethalTempo", "PressTheAttack",
    "Aftershock", "PhaseRush", "HailOfBlades", "FleetFootwork",
    "GlacialAugment", "GraspOfTheUndying", "UnsealedSpellbook"
  ),
  ID = c(
    "8465", "8214", "8112", "8128", "8010",
    "8229", "8369", "8008", "8005",
    "8439", "8230", "9923", "8021", "8351",
    "8437", "8360"
  ),
  tree = c(
    "Resolve1", "Sorcery1", "Domination1", "Domination2", "Precision1",
    "Sorcery2", "Inspiration1", "Precision2", "Precision3",
    "Resolve2", "Sorcery3", "Domination3", "Precision4", "Inspiration2",
    "Resolve3", "Inspiration3"
  )
)

```

```

)

keystone_order <- c(
  #Precision
  "Conquerer", "LethalTempo", "PressTheAttack", "FleetFootwork",
  #Domination
  "DarkHarvest", "Electrocute", "HailOfBlades",
  #Inspiration
  "GlacialAugment", "UnsealedSpellbook", "FirstStrike",
  #Sorcery
  "ArcaneComet", "SummonAery", "PhaseRush",
  #Resolve
  "GraspOfTheUndying", "Guardian", "Aftershock"
)
#Define Color Pallette for graphing and relations of color -> Rune
color_pallette <- c(
  "Precision1" = "#ffee00",
  "Precision2" = "#e2d524",
  "Precision3" = "#bbb01e",
  "Precision4" = "#8f8717",
  "Domination1" = "#af1919",
  "Domination2" = "#d81d1d",
  "Domination3" = "#811212",
  "Inspiration1" = "#267580",
  "Inspiration2" = "#36b0c0",
  "Inspiration3" = "#2c92a0",
  "Sorcery1" = "#700a8f",
  "Sorcery2" = "#8d0cb4",
  "Sorcery3" = "#5c0975",
  "Resolve1" = "#077e1b",
  "Resolve2" = "#056114",
  "Resolve3" = "#08a522"
)
rune_colors <- color_pallette[rune_stones_lookup$tree]
names(rune_colors) <- rune_stones_lookup$rune

#Filter out outliers in data & missing data entries
#Some games have extremely high damage due to non-realistic playstyles
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

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

# 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 significant
# 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
lane_stats <- TukeyHSD(model, "Lane")
#lane_stats

# Make sure factors are properly set
filtered$PrimaryKeyStone <- factor(filtered$PrimaryKeyStone)
filtered$Lane <- factor(filtered$Lane)

#Relate Keystone ID, Name, and Colors
filtered <- filtered |>
  left_join(rune_stones_lookup, by = c("PrimaryKeyStone" = "ID"))

#Reorganizes data to group by keystone order (Domination, Precision, etc.)
filtered$rune <- factor(filtered$rune, levels = keystone_order)

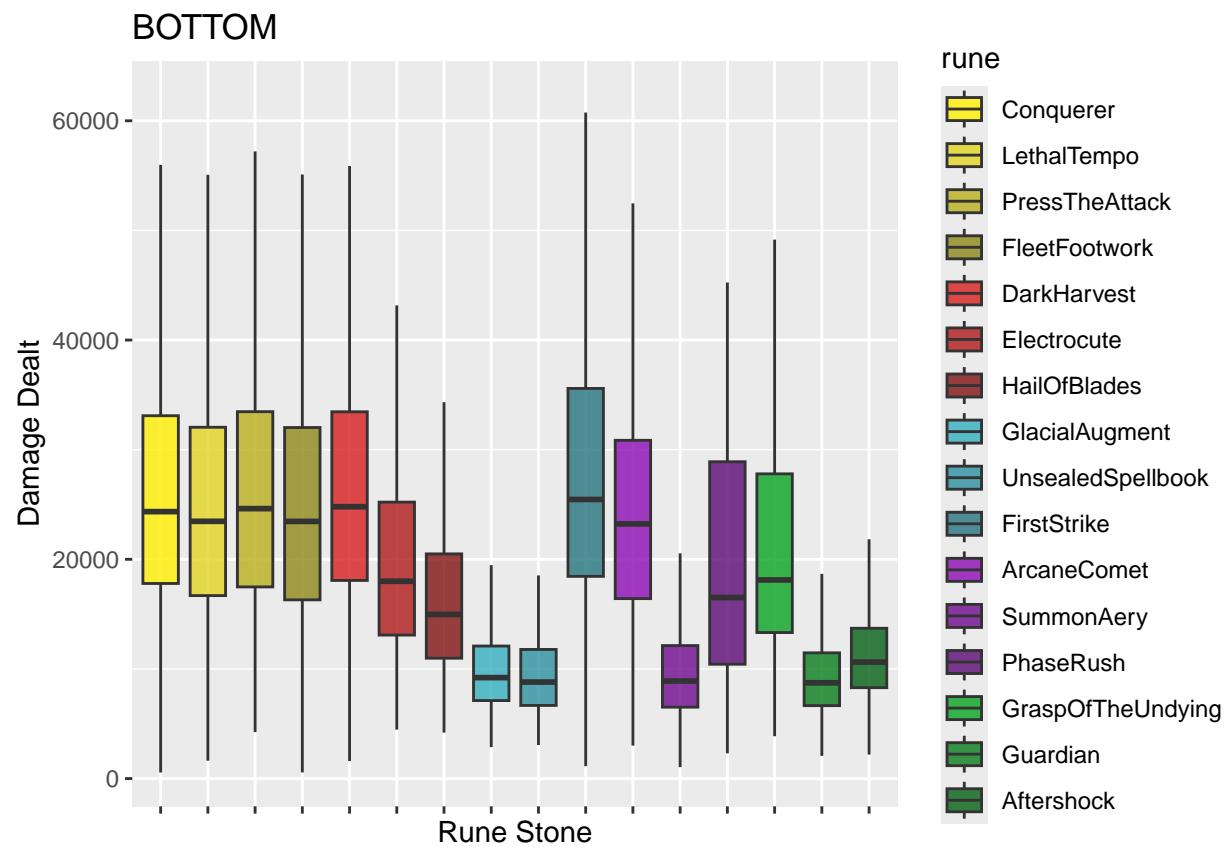
#Loop over lanes and create graph for each
lanes <- unique(filtered$Lane)
for (lane in lanes){

  df_lane <- filtered[filtered$Lane == lane, ]
  plot <- ggplot(df_lane, aes(x = rune, y = DmgDealt, fill = rune)) +
    geom_boxplot(alpha = 0.8, outlier.shape = NA) +
    scale_fill_manual(values = rune_colors) +
    xlab("Rune Stone") +
    ylab("Damage Dealt") +
    ggtitle(lane) +
    theme(axis.text.x = element_blank())

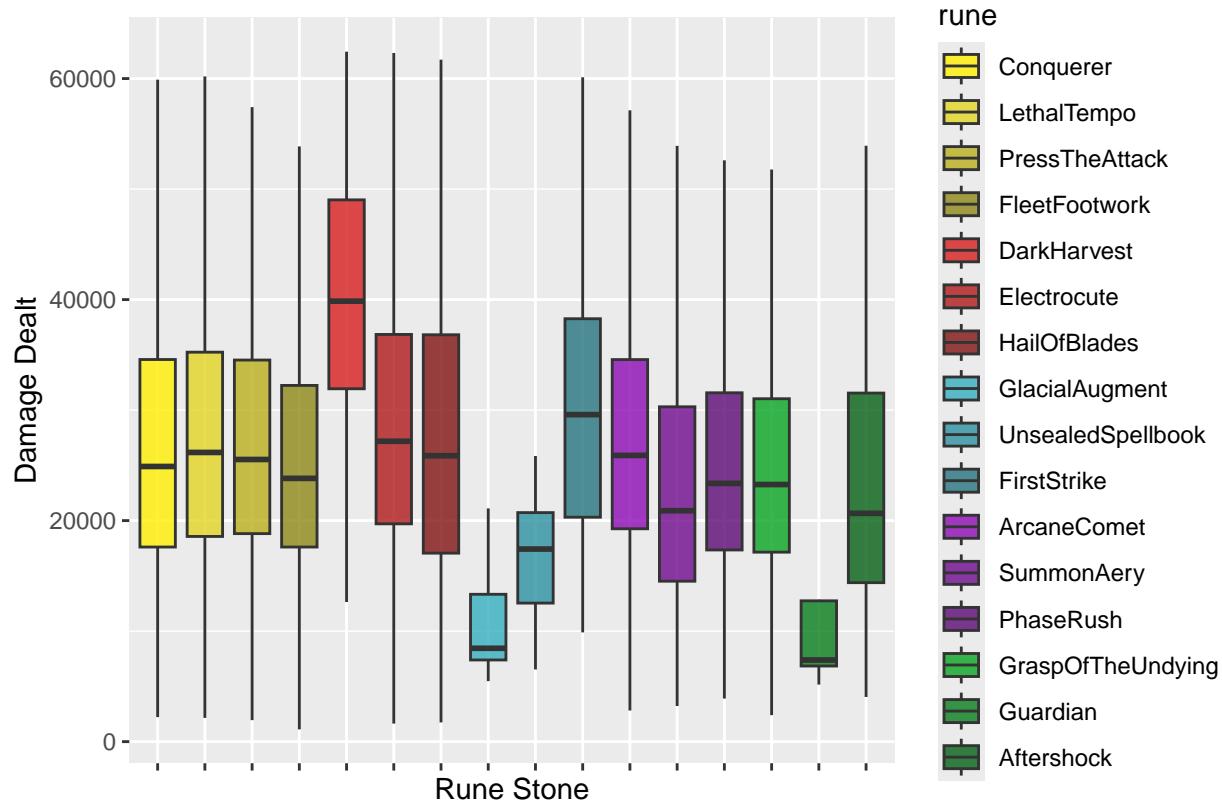
  print(plot)
}

```

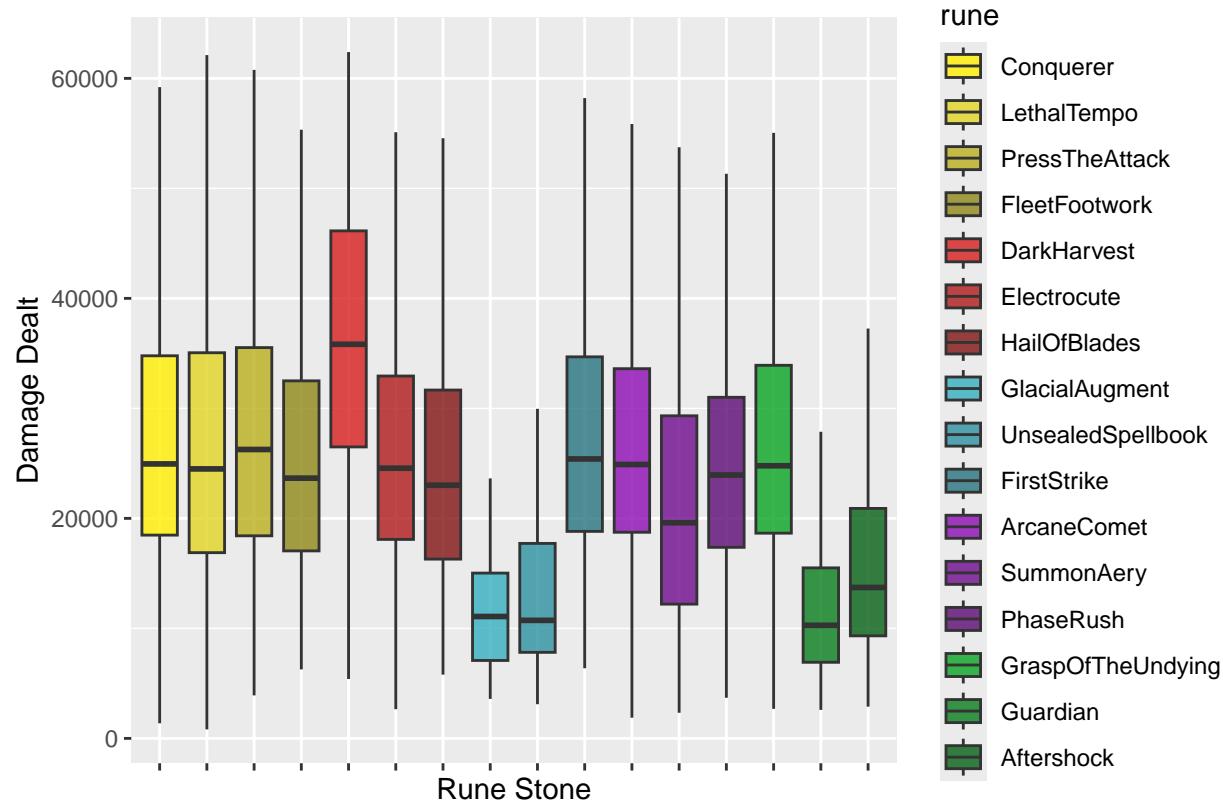
}



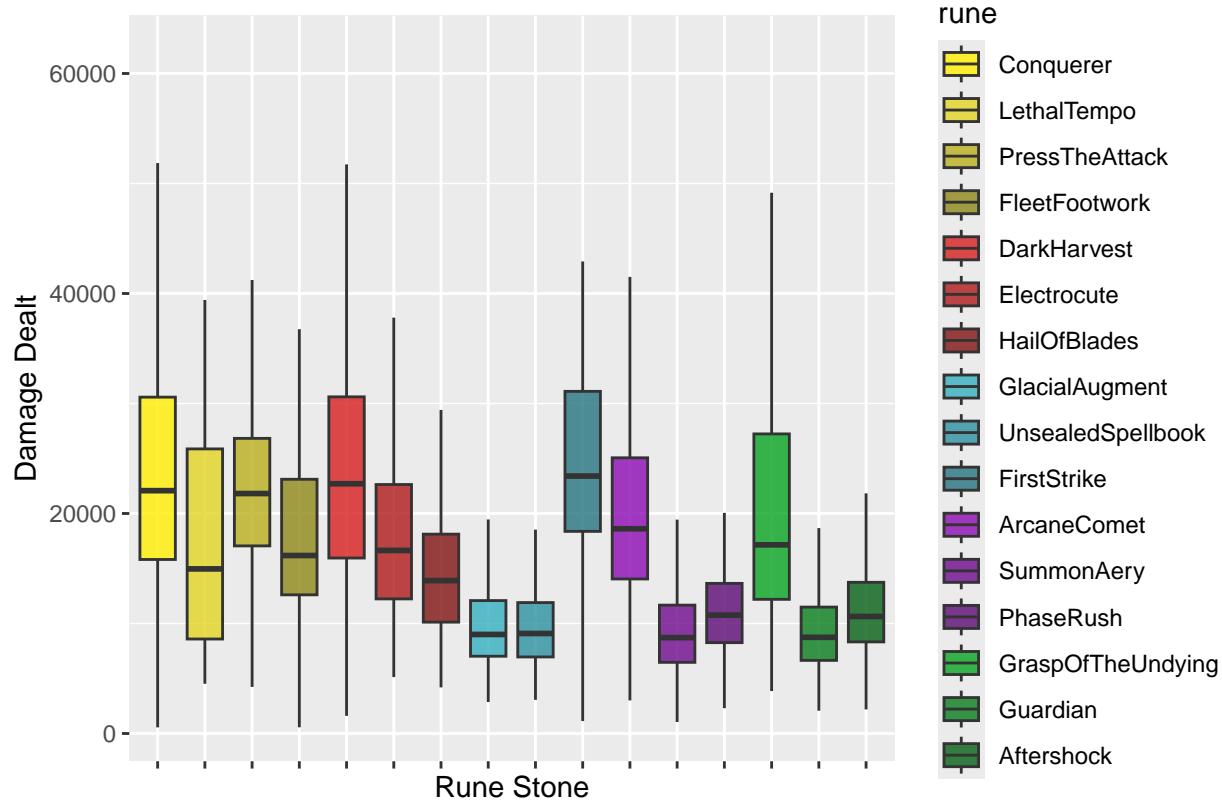
TOP



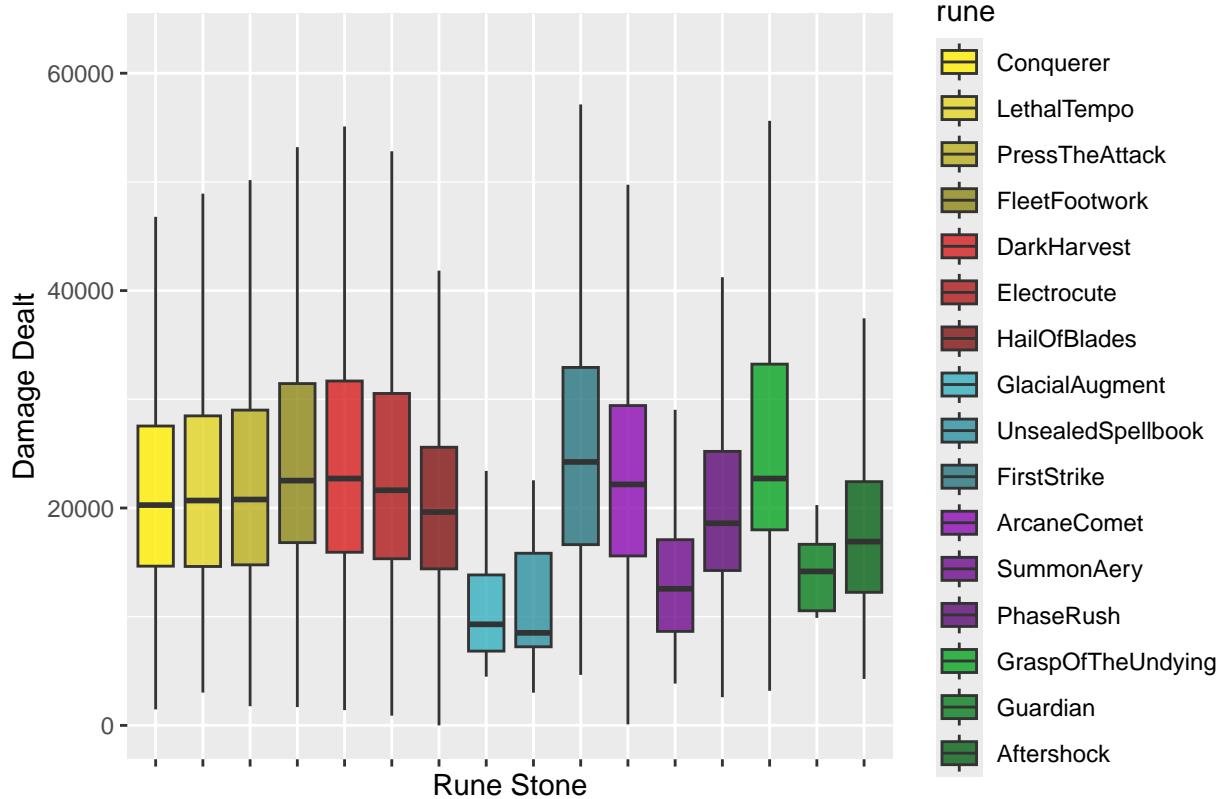
MIDDLE



SUPPORT



JUNGLE



```
#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

```

```

tab <- table(match_stats$ChampionName, match_stats$Win)
chi_square_test <- chisq.test(tab)
chi_square_test

```

```

##
## Pearson's Chi-squared test
##
## data: tab
## X-squared = 444.22, df = 170, p-value < 2.2e-16

```

```

library(ggplot2)
library(reshape2)
features <- match_stats[, c("DmgDealt", "DmgTaken",
                            "TotalGold", "kills", "deaths", "assists",

```

```

        "visionScore", "DragonKills", "BaronKills")]
}

#Scale features for clustering (Necessary when measuring distance)
scaled_features <- scale(features)

best <- NULL
best_wss <- Inf

for (i in 1:10){
  km_tmp <- kmeans(scaled_features, centers = 3, nstart = 25)
  if (km_tmp$tot.withinss < best_wss) {
    best = km_tmp
    best_wss = km_tmp$tot.withinss
  }
}

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 7525250)

km <- best # optimal run

match_stats$cluster <- km$cluster

pca <- prcomp(scaled_features)

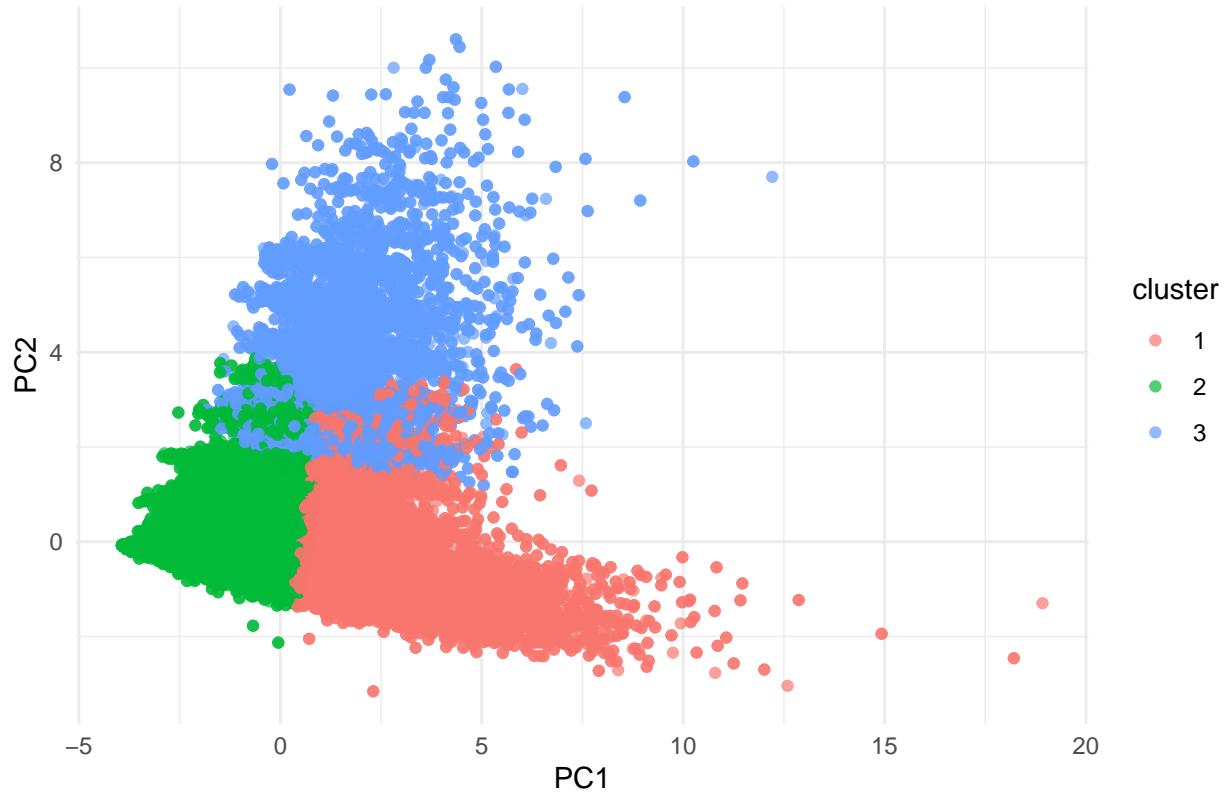
centers <- as.data.frame(km$centers)
centers$cluster <- rownames(centers)

centers_melt <- melt(centers, id = "cluster")

ggplot(data.frame(pca$x, cluster = factor(km$cluster)),
       aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  labs(title = "Player Style Clusters (PCA Projection)")

```

Player Style Clusters (PCA Projection)



```
ggplot(centers_melt, aes(x = variable, y = value, fill = cluster)) +  
  geom_bar(stat = "identity", position = "dodge") +  
  theme_minimal() +  
  labs(title = "Cluster Centers (Scaled Values)",  
       x = "Feature", y = "Scaled Value") +  
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1))
```

