Searching for Similarity: Regression

Zuhayr Ali, Bridgette Byrant, Isabelle Kirby, Rikita Patangay

2022-10-09

Regression on Steam's top 200 thousand players

This notebook will be exploring a dataset of stats of the top ranked 200 thousand players on the Steam platform (obtained from this link) https://www.kaggle.com/datasets/patrickgendotti/steam-achievementstatscom-rankings, with the goal of determining the rank of a player based on other statistics.

Cleaning data

We start by importing the dataset.

```
df <- read.csv("amended_first_200k_players.csv")
str(df)
```

```
## 'data.frame':    200010 obs. of  15 variables:
##  $ X                : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Rank             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Player.Id        : chr  "The Stranger" "lylat" "hannez" "DDtective" ...
##  $ Player.Name      : chr  "The Stranger" "lylat" "hannez" "DDtective" ...
##  $ Best.achievements: chr  "Blessing of the Flame  Requiescat  Memory Awakener  Those are too easy  I
##  $ Points           : num  132759 119589 96014 89515 88176 ...
##  $ X.               : int  153732 141645 98255 101757 95676 84083 86238 76257 67616 72712 ...
##  $ Games            : int  16755 18358 13651 12070 10338 11711 17436 4134 9402 13273 ...
##  $ X100.            : int  4493 6034 4008 3708 2127 3949 3938 1108 3266 1729 ...
##  $ All              : num  21203 28377 23981 19821 13092 ...
##  $ Badges           : int  4910 195 2009 196 645 1212 472 663 29 236 ...
##  $ XP               : int  3087192 296822 792032 60662 198956 378057 169257 139791 56210 75220 ...
##  $ Member.since     : chr  "04/15/2011" "06/03/2009" "06/23/2010" "09/01/2005" ...
##  $ Hours            : num  75592 55961 33020 9353 89603 ...
##  $ Last.update      : chr  "05/26/2022 11:41:05" "05/26/2022 11:42:01" "05/26/2022 11:40:54" "07/22/1
```

Additionally we must check for any missing values. All columns are full except for the "All" column, but due to a lack of documentation on what that column represents we will not be using that column.

```
sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##                X             Rank        Player.Id      Player.Name
##                0                0                0                0
## Best.achievements           Points               X.            Games
##                0                0                0                0
##            X100.              All           Badges               XP
##                0            22682                0                0
##     Member.since            Hours      Last.update
##                0                0                0
```

In fact, the data source does not provide any documentation on what any of the columns represent. We will thus trim the dataset to only include numeric value columns with an interpretable title.

```
df <- df[c(2,6,8,9,11,12,14)]
str(df)
```

```
## 'data.frame':    200010 obs. of  7 variables:
##  $ Rank  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Points: num  132759 119589 96014 89515 88176 ...
##  $ Games : int  16755 18358 13651 12070 10338 11711 17436 4134 9402 13273 ...
##  $ X100. : int  4493 6034 4008 3708 2127 3949 3938 1108 3266 1729 ...
##  $ Badges: int  4910 195 2009 196 645 1212 472 663 29 236 ...
##  $ XP    : int  3087192 296822 792032 60662 198956 378057 169257 139791 56210 75220 ...
##  $ Hours : num  75592 55961 33020 9353 89603 ...
```

The dataset is split by an 80/20 ratio for training and testing respectively.

```
set.seed(1010)
i <-sample(1:nrow(df), nrow(df)*0.80, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data exploration

Now for some data exploration.

```
head(train)
```

```
##          Rank  Points Games X100. Badges     XP   Hours
## 78070   78070 1278.27   307    32     34  11007  1743.3
## 64022   64022 1470.46   657    48    187  71502  1628.3
## 117885 117885  900.66  2004    31   1073 125146 11449.6
## 166987 166987  607.79   121     4     17   4603  3625.7
## 70781   70781 1370.57   305     3     95  20580  6661.2
## 190141 190141  505.28    53     0      4    934  9365.7
```

```
tail(train)
```

```
##          Rank  Points Games X100. Badges    XP   Hours
## 44802   44802 1851.76   382     3     32 19520  7592.3
## 26626   26626 2486.34   746     4     19  2872  8717.9
## 64689   64689 1460.57   405     7     22  4737  6262.6
## 169336 169336  596.22   113     1     11  2817  3644.4
## 141477 141477  742.92  1005     1     81 13169  7146.0
## 23876   23876 2634.24   994    48     86 16760 14110.3
```

```
range(train$Points)
```

```
## [1]    467.6 132759.3
```

```
mean(train$Points)
```

```
## [1] 1535.943
```

```
median(train$Points)
```

```
## [1] 1049.54
```

```
range(train$Games)
```

```
## [1]     1 23224
```

```
mean(train$Games)
```

```
## [1] 538.7729
```

```
median(train$Games)
```

```
## [1] 285
```

```
range(train$Badges)
```

```
## [1]     0 19496
```

```
mean(train$Badges)
```

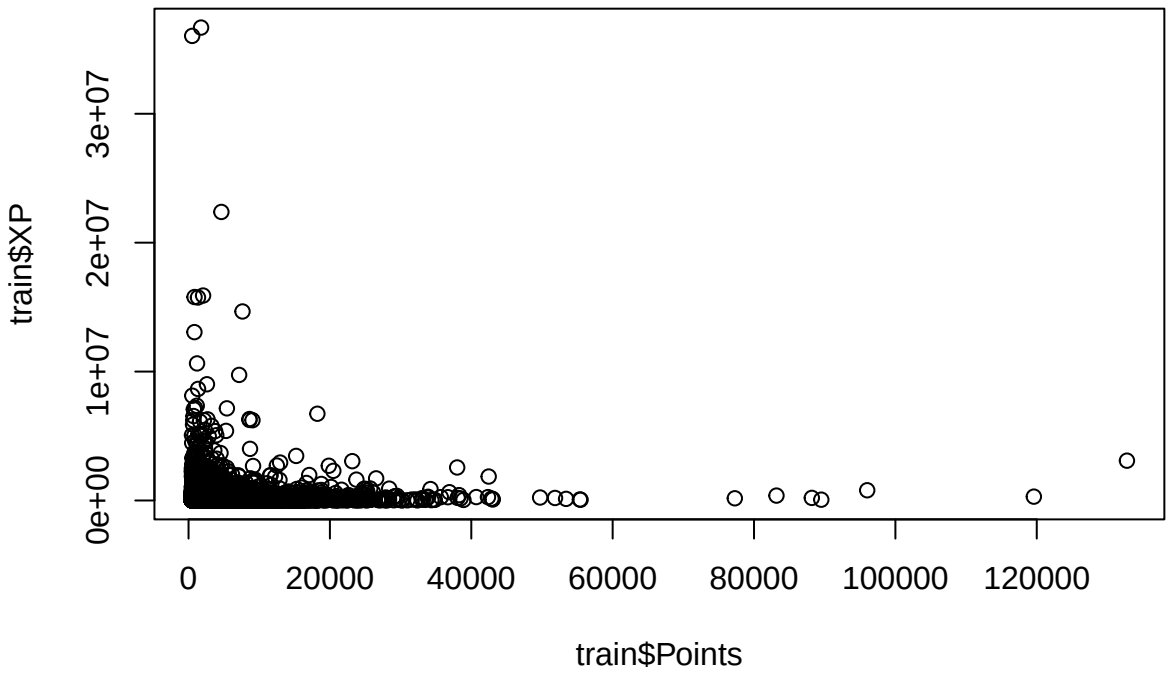
```
## [1] 84.07015
```

```
median(train$Badges)
```

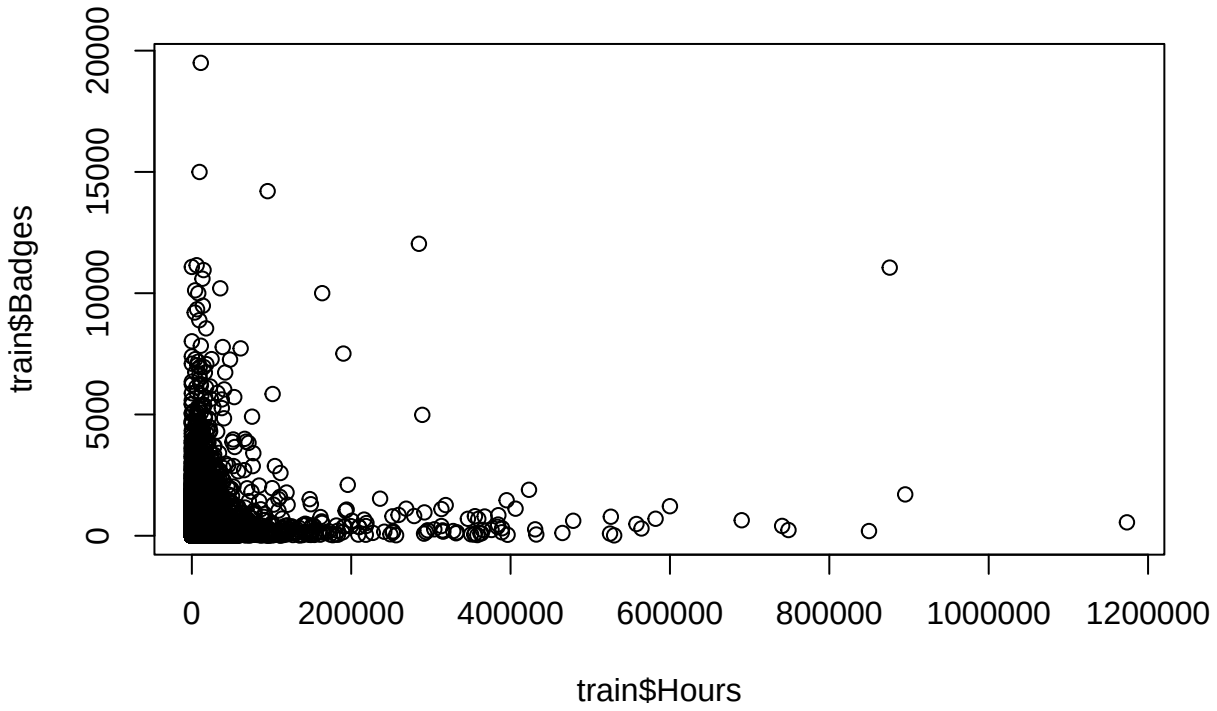
```
## [1] 33
```

Here are also some plots to compare the correlation of factors.

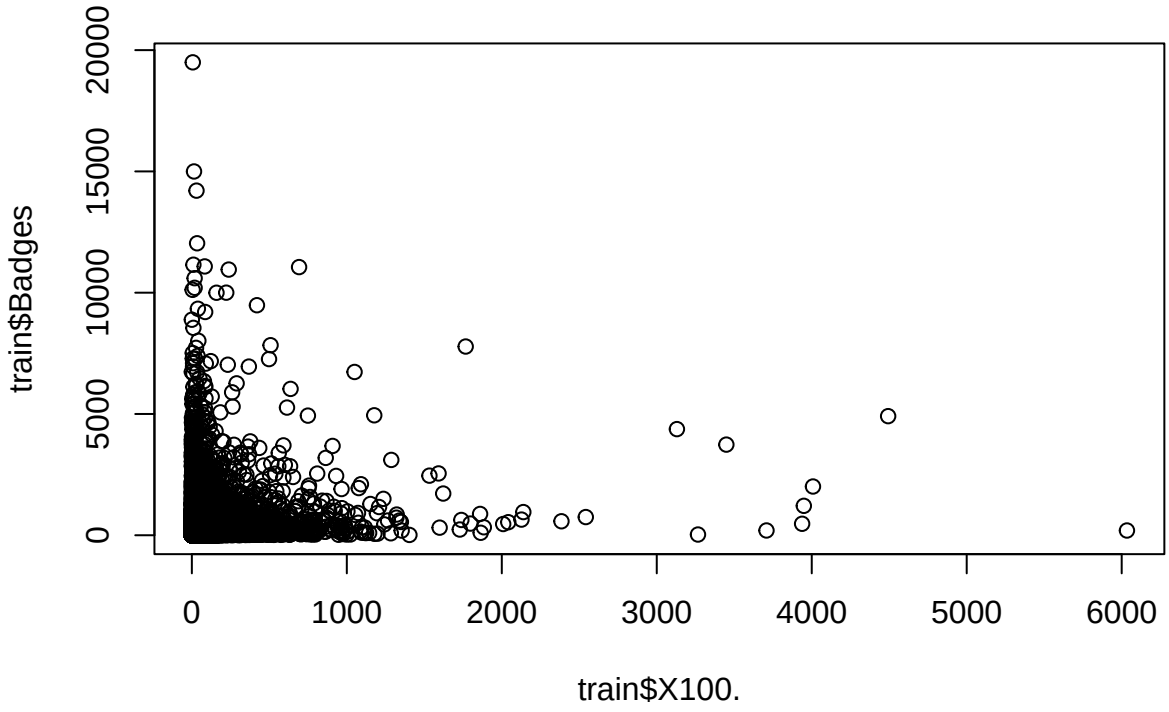
```
plot(train$Points, train$XP)
```



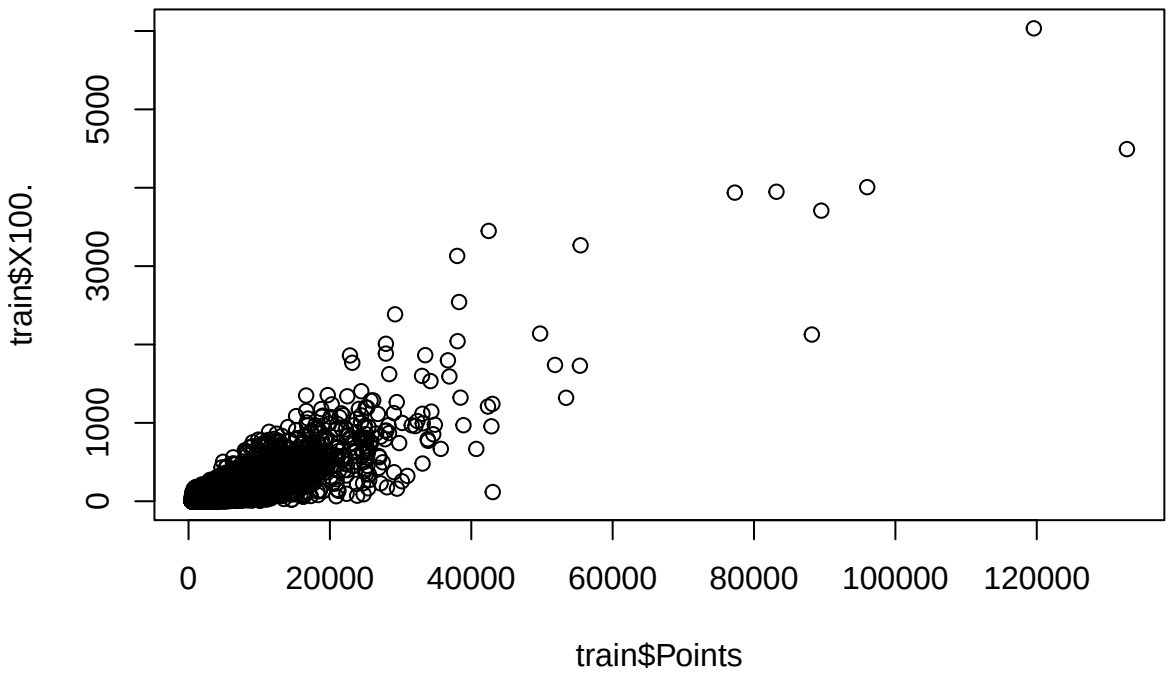
```
plot(train$Hours, train$Badges)
```



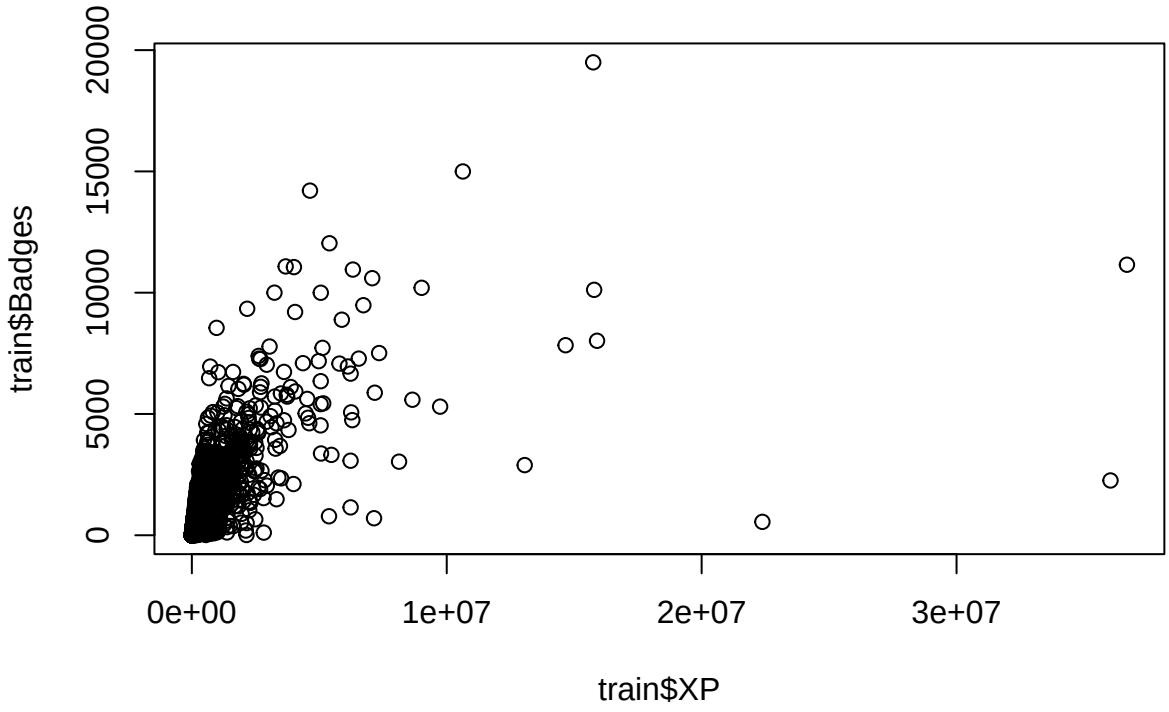

```
plot(train$X100., train$Badges)
```



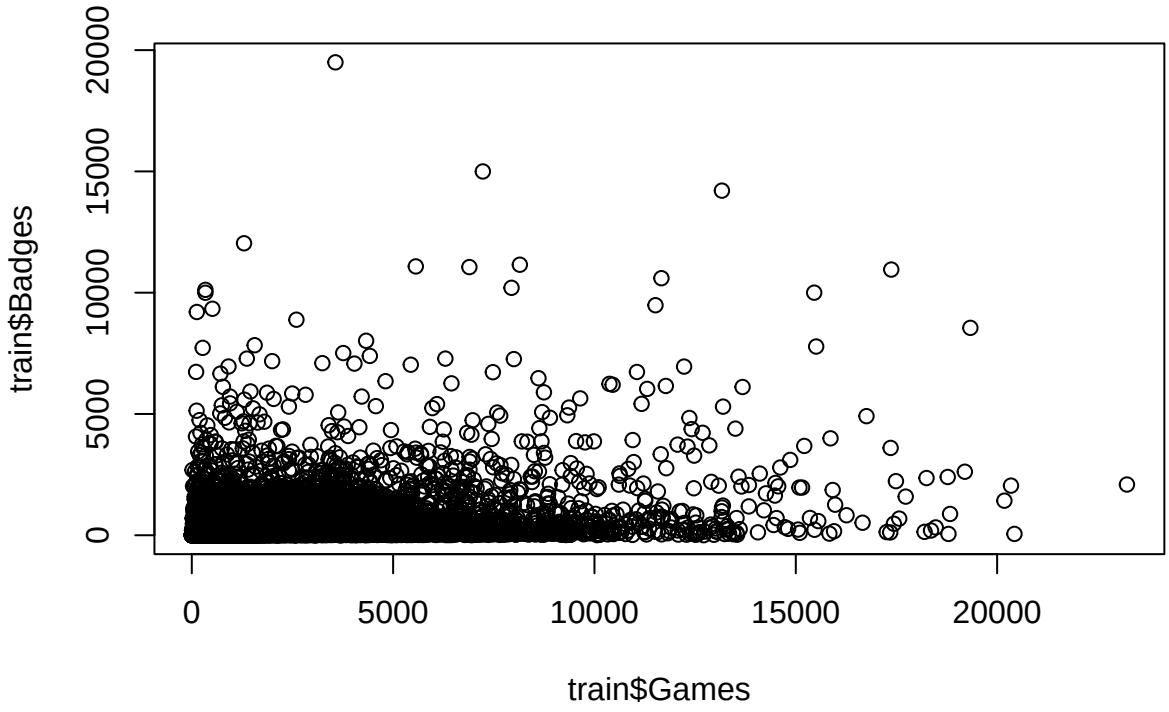

```
plot(train$Points, train$X100.)
```




```
plot(train$XP, train$Badges)
```




```
plot(train$Games, train$Badges)
```



Computing the correlation matrix of all columns shows us that most of the variables don't have a strong correlation with each other - only the pair of "Points" and "x100." columns showed an absolute correlation higher than 0.7 with approximately 0.83, while the next two strongest correlations are approximately 0.67 and -0.6.

```
cor(df)
```

```
##               Rank      Points      Games      X100.     Badges          XP
## Rank    1.00000000 -0.60480117 -0.3220264 -0.3568166 -0.1483899 -0.06941368
## Points -0.60480117  1.00000000  0.4097622  0.8293519  0.1990152  0.09297835
## Games  -0.32202636  0.40976225  1.0000000  0.3380270  0.4089448  0.20280234
## X100.  -0.35681655  0.82935187  0.3380270  1.0000000  0.2028438  0.10089669
## Badges -0.14838985  0.19901522  0.4089448  0.2028438  1.0000000  0.67018946
## XP     -0.06941368  0.09297835  0.2028023  0.1008967  0.6701895  1.00000000
## Hours  -0.17922432  0.21652835  0.1772684  0.1590048  0.1580455  0.09533649
##              Hours
## Rank   -0.17922432
## Points  0.21652835
## Games   0.17726840
## X100.   0.15900481
## Badges  0.15804552
## XP      0.09533649
## Hours   1.00000000
```

We will now train 3 different types of regression models on the dataset - linear regression, kNN regression, and decision tree regression.

Linear regression

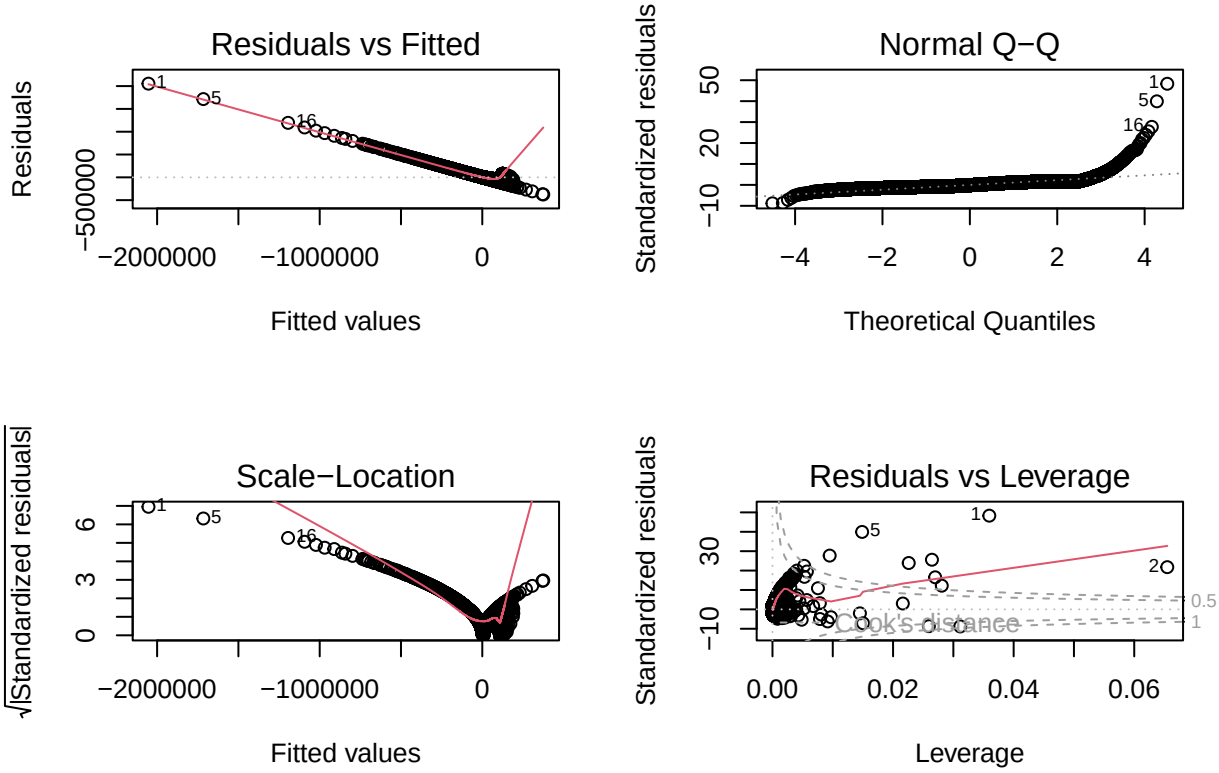
Training

```
lm1 <- lm(Rank~Points+X100., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = Rank ~ Points + X100., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -375138  -34554   -5840   32235 2053174
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.401e+05  1.594e+02   878.7   <2e-16 ***
## Points      -3.227e+01  1.077e-01  -299.8   <2e-16 ***
## X100.        4.655e+02  3.181e+00   146.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43290 on 160005 degrees of freedom
## Multiple R-squared:  0.4377, Adjusted R-squared:  0.4377
## F-statistic: 6.227e+04 on 2 and 160005 DF,  p-value: < 2.2e-16
```

Residual plots

```
par(mfrow=c(2,2))
plot(lm1)
```



Testing and evaluation

```
pred1 <- predict(lm1, newdata=test)

cor1 <- cor(pred1, test$Rank)
mse1 <- mean((pred1-test$Rank)^2)
rmse1 <- sqrt(mse1)

print(paste('correlation:', cor1))
```

```
## [1] "correlation: 0.643817838741176"
```

```
print(paste('mse:', mse1))
```

```
## [1] "mse: 1959246688.37052"
```

```
print(paste('rmse:', rmse1))
```

```
## [1] "rmse: 44263.3786370914"
```

kNN regression

Normalization

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
df_norm <- as.data.frame(lapply(df[, 1:7], normalize))
head(df_norm)
```

```
##           Rank    Points     Games     X100.     Badges          XP       Hours
## 1 0.000000e+00 1.0000000 0.6742866 0.7446139 0.25184653 0.084152426 0.064415394
## 2 4.999775e-06 0.9004446 0.7388015 1.0000000 0.01000205 0.008090942 0.047686793
## 3 9.999550e-06 0.7222422 0.5493621 0.6642360 0.10304678 0.021589656 0.028137924
## 4 1.499933e-05 0.6731171 0.4857327 0.6145177 0.01005334 0.001653559 0.007970054
## 5 1.999910e-05 0.6629912 0.4160261 0.3525025 0.03308371 0.005423255 0.076354924
## 6 2.499888e-05 0.6251925 0.4712843 0.6544581 0.06216660 0.010305292 0.048589985
```

```
i <-sample(1:nrow(df), nrow(df)*0.80, replace=FALSE)
train_norm <- df_norm[i,]
test_norm <- df_norm[-i,]
```

Training, testing, and evaluation

```
library(caret)
```

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

```
## Loading required package: lattice
```

```
cor_k <- rep(0, 10)
mse_k <- rep(0, 10)
i <- 1
for (k in seq(1, 19, 2)){
fit_k <- knnreg(train_norm[,2:7],train_norm[,1], k=k)
pred_k <- predict(fit_k, test_norm[,2:7])
cor_k[i] <- cor(pred_k, test_norm$Rank)
mse_k[i] <- mean((pred_k - test_norm$Rank)^2)
i <- i + 1
}

print(paste('max cor_k:', 2*which.max(cor_k)-1))
```

```
## [1] "max cor_k: 7"
```

```
print(paste('min mse_k:', 2*which.min(mse_k)-1))
```

```
## [1] "min mse_k: 7"
```

```
k_index <- (which.max(cor_k)+which.min(mse_k))/2

print(paste('correlation:', cor_k[k_index]))
```

```
## [1] "correlation: 0.994480389863142"
```

```
print(paste('mse:', mse_k[k_index]))
```

```
## [1] "mse: 0.000922313638812161"
```

Decision tree regression

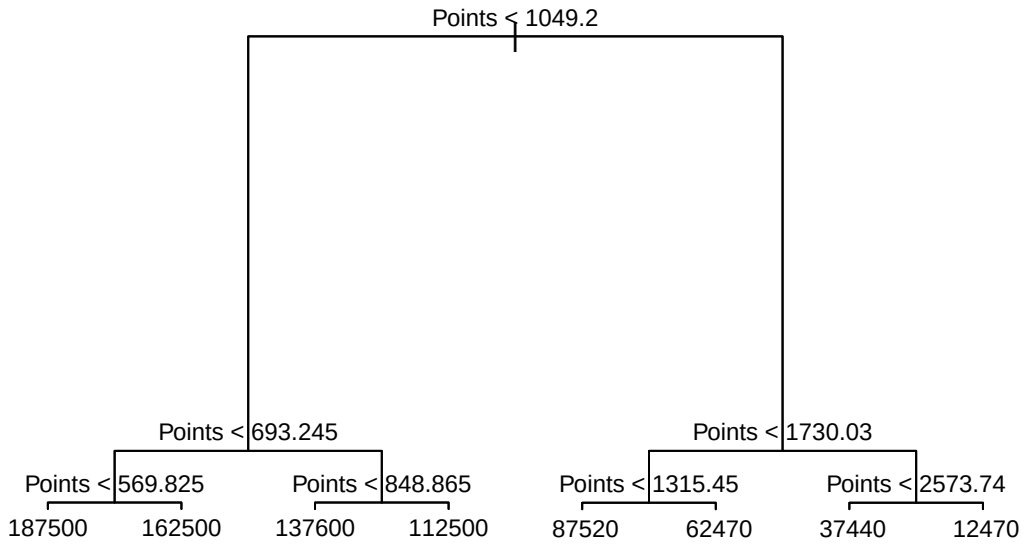
Training and tree formation

```
library(tree)

tree1 <- tree(Rank~Points+X100., data=train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = Rank ~ Points + X100., data = train)
## Variables actually used in tree construction:
## [1] "Points"
## Number of terminal nodes:  8
## Residual mean deviance:  5.2e+07 = 8.32e+12 / 160000
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -12540.000  -6236.000     -4.887      0.000   6243.000  12540.000
```

```
plot(tree1)
text(tree1, cex=0.75, pretty=0)
```



Testing and evaluation

```
pred3 <- predict(tree1, newdata=test)

print(paste('correlation:', cor(pred3, test$Rank)))
```

```
## [1] "correlation: 0.992113495557268"
```

```r
rmse_tree <- sqrt(mean((pred3-test$Rank)^2))
print(paste('rmse:', rmse_tree))
```

```
## [1] "rmse: 7242.4839552329"
```