Disc Golf Aging Curves Final Project

Lincoln Stewart

May 25, 2025

Contents

List of Figures

List of Tables


```
library(ggplot2)
library(knitr)
library(dplyr)
mean_SS_t_mod <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc
mean_SS_t <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Gol

mean_SS_t <- mean_SS_t %>%
  select(-X) %>%
  rename(DF_Used = NA.)

mean_SS_t_mod <- mean_SS_t_mod %>%
  select(-X)

clustdm_cv <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Go

clustdm <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Golf-



MPO_ar_pt <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Gol
```

```
clss <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Golf-Agi
clss_sc <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Golf-
clss_hc <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-Golf-
clss_kmeans <- read.csv("C:/Users/lincs/OneDrive/Documents/Independent Study/github/Disc-G
```

Overview and Description of Data

The goal of this independent study has been to figure out whether or not it is more effective to predict players skill level using generic aging curves across all athletes or to have position specific ones. The sport in which we used for our research was disc golf. The data that we would use was from the PDGA website that would include things like players' ratings, names and ages, and their stats from when they have competed. Using a combination of all this different data is how the research was conducted.

Method of determinig player type

Since one of the biggest determiners of this study was the way players were grouped and therefore assigned player types. This was achieved through multiple methods of clustering. The kind that were was hierarchical, GMM, spectral, and k-means. Each different clustering method used different methods of clustering, and would look at 2 clusters through 6 for each method for each kind of clustering. This gave us a total of 65 different ways in which the players were grouped.

Description of model making

After grouping the players into different types with the use of different clustering methods, it was then time to predict whether or not the player type model was more effective or the non player type. The model was then split into two different ones, one for player type and one for a regular generic one. The models fitted are as follows.

$outp < -lm(pt_ratr\ pt_ttype * pt_aages + pt_ttype * I(pt_aages^2), data = MPO_ar_pt_train2)$

$out < -lm(ratr\ ages + I(ages^2), data = MPO_ar_pt_train)$

The first being the player type model, the second being the generic model. From there, different loops were run through with multiple different iterations. The parameters that were used to then measure the fit and effectiveness of the model was adjusted R^2, sum squared error, and mean squared error. This was all put into a table titled mean_SS_t for spline model and mean_SS_t_mod for the quadratic model.

Results

Quadratic model

The use of the quadratic model was the original vision for the the project. This was the most general use to explore the effect of aging curves for player type and non player type models. The model that wads fitted for the player type is as follows

$outp < -lm(pt_ratr\ pt_type * pt_ages + pt_type * I(pt_ages^2), data = MPO_ar_pt_train2)$

and the non player type

$out < -lm(ratr\ ages + I(ages^2), data = MPO_ar_pt_train)$

Visualizations

```
# Best player type
names(MPO_ar_pt)[which.min(mean_SS_t_mod$Mean_MSE_PlayerType) + 12]
```

```
## [1] "X3_anovadot"
```

```
best_order <- names(MPO_ar_pt)[order(mean_SS_t_mod$Mean_MSE_PlayerType) + 12]

mean_SS_t_mod[order(mean_SS_t_mod$Mean_MSE_PlayerType), ]
```

```
##    Mean_SSE_PlayerType Mean_SSE_NoPlayerType Mean_MSE_NoPlayerType
## 22             1812967               2029109              27.62017
## 7              1808377               2029321              27.68965
## 12             1829028               2059000              27.84625
## 17             1829094               2029359              27.65951
## 27             1828267               2042691              27.78894
## 3              1852888               2049072              27.78167
## 2              1875934               2039958              27.73684
## 51             1876042               2016330              27.60372
```

## 48	1894022	2016825	27.61412
## 35	1894683	2038657	27.72395
## 1	1897019	2041178	27.71080
## 64	1902035	2056631	27.85466
## 59	1898223	2026952	27.67681
## 58	1891780	2016385	27.64847
## 46	1906260	2044174	27.75781
## 29	1903795	2024075	27.71386
## 19	1906257	2017637	27.65140
## 53	1902101	2015484	27.65038
## 61	1907646	2049157	27.83257
## 63	1910397	2052516	27.84625
## 62	1933400	2030904	27.69415
## 52	1929263	2041628	27.77066
## 5	1924179	2068624	27.91430
## 26	1918246	2044407	27.78155
## 6	1923303	2042154	27.72829
## 23	1929491	2044212	27.76277
## 56	1918346	2049437	27.84200
## 18	1928856	2043675	27.78147
## 13	1930196	2042087	27.75051
## 21	1920911	2053994	27.85957
## 14	1924897	2028434	27.73304
## 11	1925972	2041854	27.78511
## 16	1932213	2060688	27.87307
## 49	1946911	2042641	27.74339
## 28	1935173	2034507	27.73582
## 9	1935257	2030883	27.74946
## 42	1958380	2020002	27.63328
## 54	1944021	2020781	27.63935
## 4	1953874	2047928	27.80660
## 8	1985392	2035325	27.74180
## 45	1985834	2021756	27.58709
## 47	2576619	2033476	27.71304
## 44	1995036	2032091	27.63938
## 57	2593895	2039833	27.77220
## 43	2027473	2064729	27.84851

```
## 55            2014763              2026295             27.63485
## 33            2014004              2015357             27.53953
## 50            2042224              2057246             27.87562
## 32            2045730              2045595             27.67222
## 65            2044370              2045306             27.85012
## 25            2055500              2020654             27.59924
## 38            2046037              2045939             27.74480
## 34            2055059              2056591             27.77994
## 37            2058148              2057991             27.78907
## 15            2072885              2029433             27.67812
## 30            2084988              2012781             27.57460
## 10            2075215              2035967             27.79300
## 60            2104821              2042535             27.74894
## 20            2089286              2029806             27.75953
## 39            2247448              2041211             27.69010
## 24            2127114              2027008             27.60307
## 40            2343757              2049389             27.75514
## 31                 NA                   NA                   NA
## 36                 NA                   NA                   NA
## 41                 NA                   NA                   NA
##    Mean_MSE_PlayerType Adj_R_Sqr_NoPlayerType Adj_R_Sqr_PlayerType
## 22            26.05634              0.3020976            0.4277503
## 7             26.09116              0.3012010            0.4266341
## 12            26.18672              0.3020962            0.4267239
## 17            26.19032              0.3007004            0.4271734
## 27            26.24235              0.3014891            0.4273772
## 3             26.32618              0.3002262            0.4250567
## 2             26.56358              0.3014817            0.4058088
## 51            26.60677              0.3018588            0.3645871
## 48            26.67504              0.3004638            0.3952138
## 35            26.68417              0.3002519            0.4021110
## 1             26.69381              0.3020046            0.3673521
## 64            26.73114              0.3016422            0.4236578
## 59            26.73664              0.3019483            0.4177411
## 58            26.74484              0.3000259            0.3954371
## 46            26.79032              0.3012664            0.3645132
## 29            26.81155              0.2999594            0.4235662
```

```
## 19          26.81472       0.3001964       0.4239551
## 53          26.82159       0.2992322       0.3941634
## 61          26.83916       0.3006623       0.3639672
## 63          26.85032       0.3000592       0.3949741
## 62          26.85111       0.3014819       0.4185866
## 52          26.86534       0.3013249       0.4180589
## 5           26.88534       0.3017461       0.4291675
## 26          26.89327       0.3017002       0.3582859
## 6           26.89765       0.3002870       0.3577969
## 23          26.90537       0.3012105       0.4178517
## 56          26.91900       0.3011061       0.3647556
## 18          26.92138       0.3015478       0.4210399
## 13          26.92804       0.3018620       0.4184414
## 21          26.93433       0.3015446       0.3580143
## 14          26.95386       0.2992421       0.4240148
## 11          26.96421       0.3015837       0.3587028
## 16          26.98166       0.3021914       0.3587873
## 49          26.99309       0.3012747       0.4227192
## 28          26.99732       0.3018207       0.4196219
## 9           27.02549       0.2984856       0.4235851
## 42          27.02663       0.3015972       0.4090414
## 54          27.03613       0.3002943       0.4262173
## 4           27.05945       0.3015044       0.4082551
## 8           27.07540       0.3016094       0.4218728
## 45          27.12552       0.2997127       0.4169413
## 47          27.17699       0.3009915       0.4229920
## 44          27.20483       0.2998314       0.4176743
## 57          27.28113       0.3008193       0.4175093
## 43          27.38725       0.3017841       0.4095302
## 55          27.39579       0.3007613       0.4246947
## 33          27.53224       0.2987421       0.3116975
## 50          27.61847       0.3016259       0.4317560
## 32          27.67315       0.3014976       0.3024693
## 65          27.71179       0.3003505       0.4343878
## 25          27.74055       0.3000908       0.4030884
## 38          27.74552       0.3007732       0.3017439
## 34          27.77166       0.3000493       0.3130368
```

```
## 37          27.79019          0.3012552          0.3022306
## 15          27.87069          0.2990087          0.4023864
## 30          27.95591          0.2992221          0.4035353
## 10          27.96177          0.2991061          0.4026315
## 60          28.03573          0.3016895          0.4298452
## 20          28.06582          0.2992805          0.4060509
## 39          28.17876          0.3020104          0.3252229
## 24          28.23518          0.2999217          0.3473182
## 40          28.49555          0.3011804          0.3242155
## 31                NA                 NA                 NA
## 36                NA                 NA                 NA
## 41                NA                 NA                 NA
```

```r
## table of best clustering types
tab_best_order <- data.frame(
  Best_cluster_types = c(best_order[1:6])
)

kable(tab_best_order)
```

Table 1: Table showing best clustering types

Best_cluster_types
X3_anovadot
X3_rbfdot
X3_polydot
X3_laplacedot
X3_splinedot
nc4

```r
# Visualizing results
pp <- apply(clustdm_cv[, c(2, 3, 5, 6)], 2, function(x) tapply(x, clustdm_cv$X3_anovadot,
barplot(t(pp), beside = TRUE, ylab = "Standardized Statistic (C1,C2, C1P, C2P)", xlab = "F
```

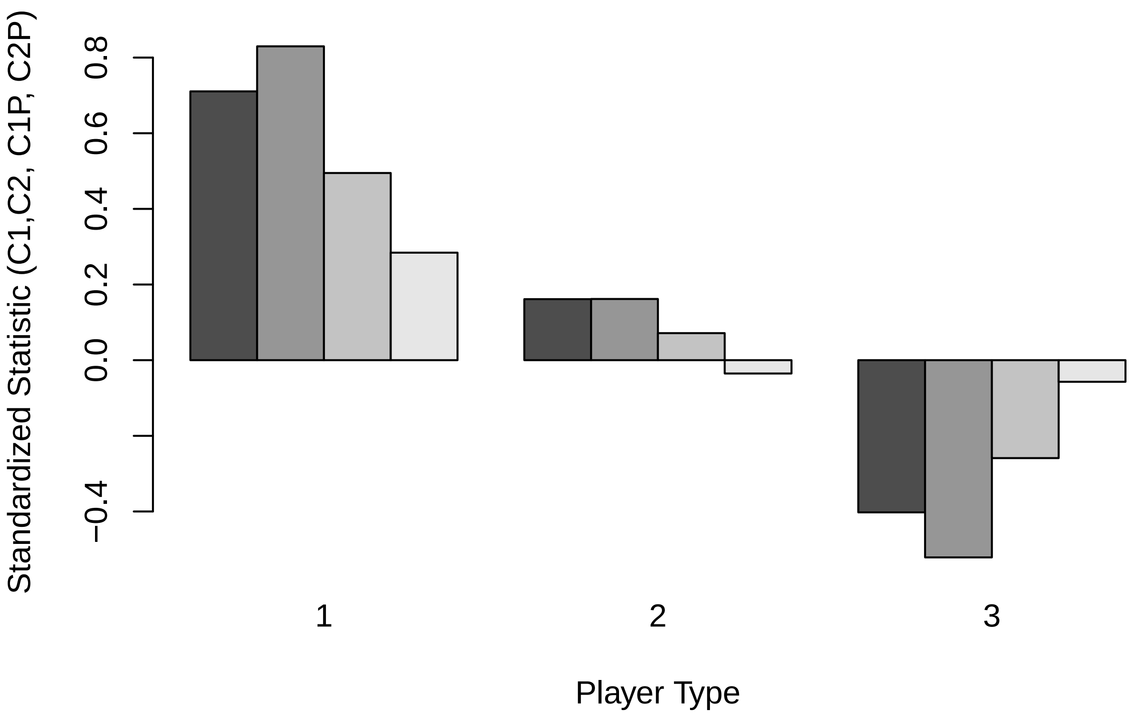

Figure 1: Bar plot of showing spread of player data across different clusters

```r
# Final polynomial-based visualization for sanity check
out <- lm(ratr ~ -1 + factor(X3_anovadot) + factor(X3_anovadot) / ages + factor(X3_anovad
          data = MPO_ar_pt)
summary(out)
```

```
##
## Call:
## lm(formula = ratr ~ -1 + factor(X3_anovadot) + factor(X3_anovadot)/ages +
##     factor(X3_anovadot)/I(ages^2), data = MPO_ar_pt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -243.459  -10.402    1.074   14.676   67.349
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## factor(X3_anovadot)1            1034.5997     0.4731 2186.66   <2e-16 ***
## factor(X3_anovadot)2            1015.6658     0.4071 2494.99   <2e-16 ***
## factor(X3_anovadot)3             995.9549     0.8568 1162.43   <2e-16 ***
## factor(X3_anovadot)1:ages          8.2355     0.6081   13.54   <2e-16 ***
## factor(X3_anovadot)2:ages         16.3652     0.4344   37.67   <2e-16 ***
## factor(X3_anovadot)3:ages         22.6632     1.2025   18.85   <2e-16 ***
## factor(X3_anovadot)1:I(ages^2)  -25.3701     0.5585  -45.43   <2e-16 ***
## factor(X3_anovadot)2:I(ages^2)  -17.9346     0.4200  -42.71   <2e-16 ***
## factor(X3_anovadot)3:I(ages^2)  -15.8406     1.1498  -13.78   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.94 on 13285 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9994
## F-statistic: 2.397e+06 on 9 and 13285 DF,  p-value: < 2.2e-16
```

```r
ages_plot <- seq(-1, 1, 0.1)

rat_pred1 <- coef(out)[1] + coef(out)[4] * ages_plot + coef(out)[7] * ages_plot^2
rat_pred2 <- coef(out)[2] + coef(out)[5] * ages_plot + coef(out)[8] * ages_plot^2
rat_pred3 <- coef(out)[3] + coef(out)[6] * ages_plot + coef(out)[9] * ages_plot^2
```

```r
ages_plot2 <- ages_plot * sd(MPO_ar_pt$Age) + mean(MPO_ar_pt$Age)

plot(rat_pred2 ~ ages_plot2, type = "l", ylim = c(min(rat_pred3), max(rat_pred1)), lwd = 2
points(rat_pred1 ~ ages_plot2, type = "l", col = "blue", lwd = 2)
points(rat_pred3 ~ ages_plot2, type = "l", col = "red", lwd = 2)
```

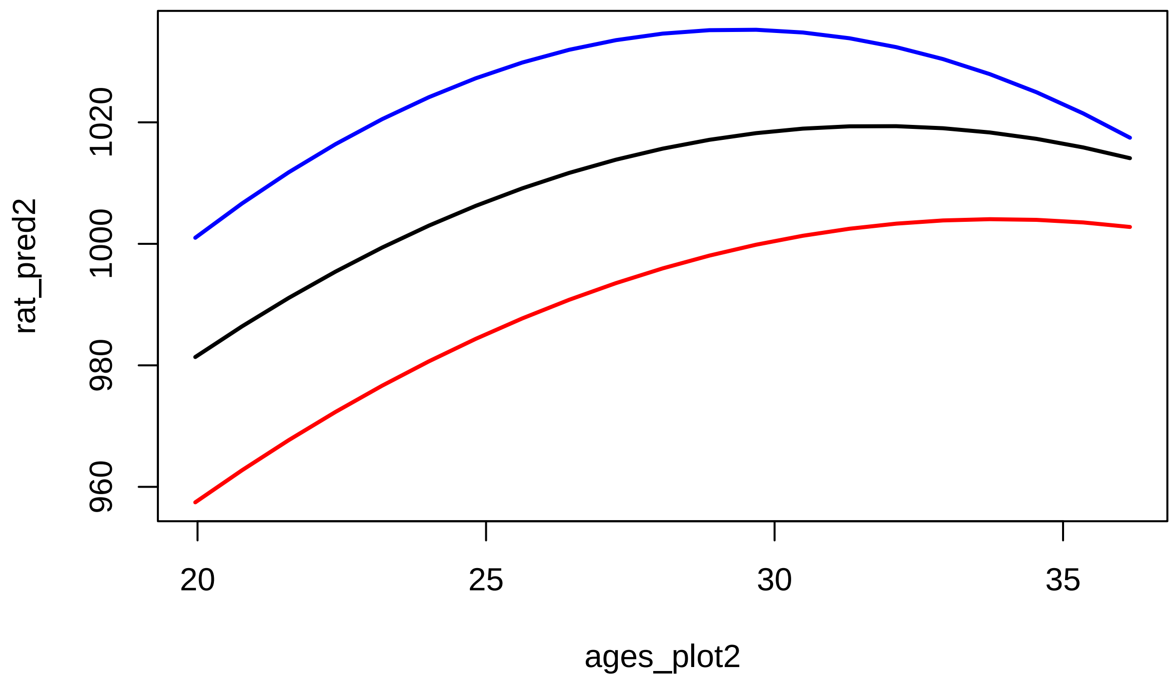

Figure 2: Different polynomials from the clusters

These are a couple visualizations of of scatter plots of all the players.

```r
## scatter plot with different color points to show
ggplot(MPO_ar_pt, aes(x = Age, y = ratr)) +
  geom_point(alpha = 0.3) +
  labs(title = "All Players' Ratings by Age",
       x = "Age", y = "Player Rating") +
  theme_minimal()
```

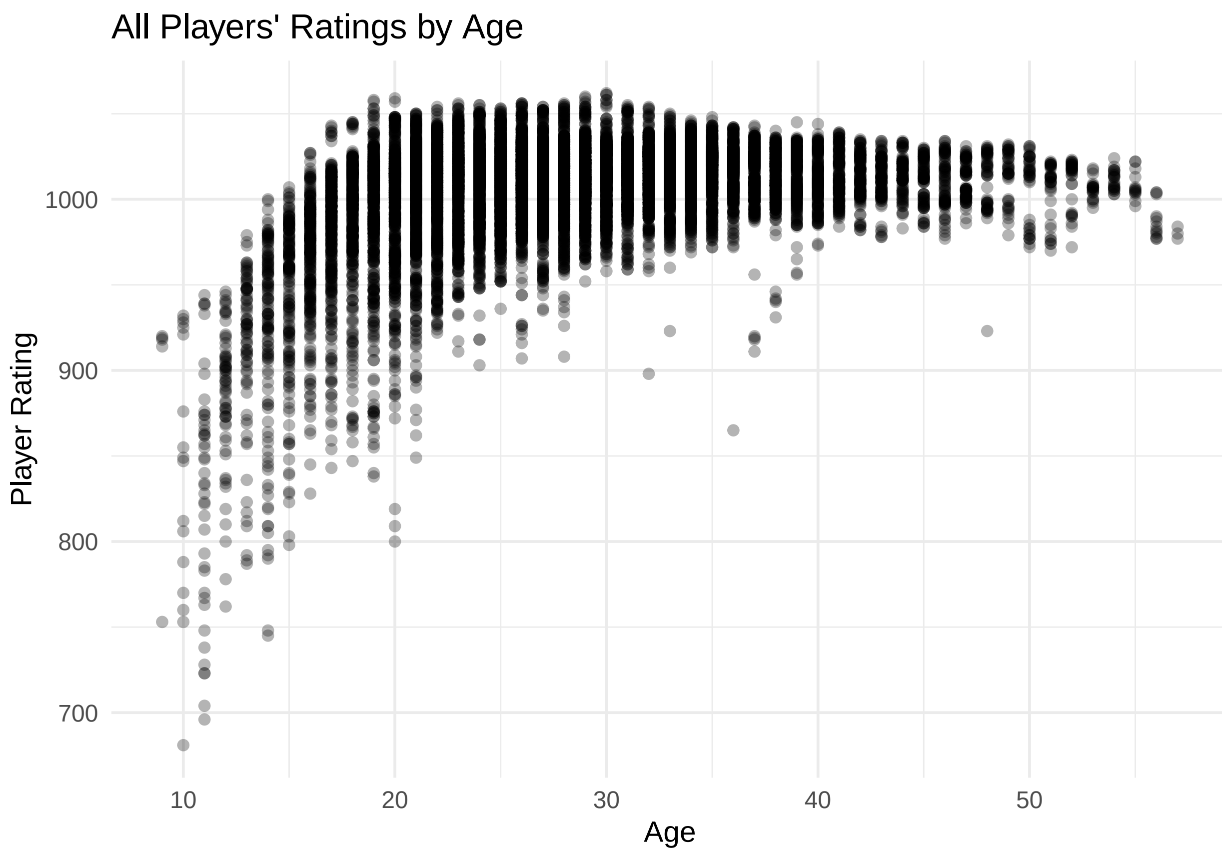

Figure 3: SCatter plot of all players and their ratings

```r
# with color
ggplot(MPO_ar_pt, aes(x = Age, y = ratr, color = as.factor(`X3_anovadot`))) +
  geom_point(alpha = 0.4) +
  labs(title = "Ratings by Age and Player Type",
       x = "Age", y = "Player Rating", color = "Player Type") +
  theme_minimal()
```

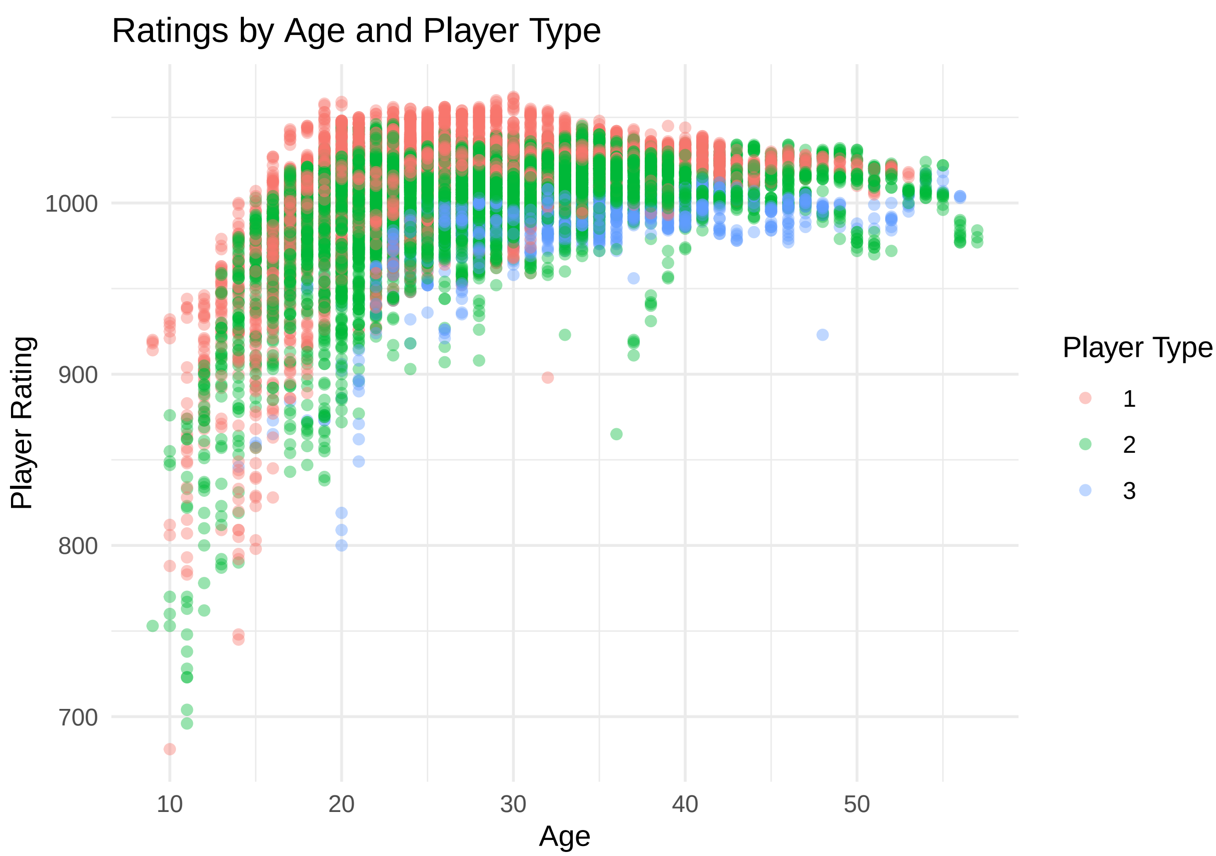

Figure 4: Scatter plot seperated by cluster type, seperated by color

```r
# using facet wrap
ggplot(MPO_ar_pt, aes(x = Age, y = ratr, color = as.factor(`X3_anovadot`))) +
  geom_point(alpha = 0.4) +
  facet_wrap(~`X3_anovadot`) +
  labs(title = "Ratings by Age and Player Type",
       x = "Age", y = "Player Rating", color = "Player Type") +
  theme_minimal()
```

This is a representation of all the different clustering methods used and their effectiveness.

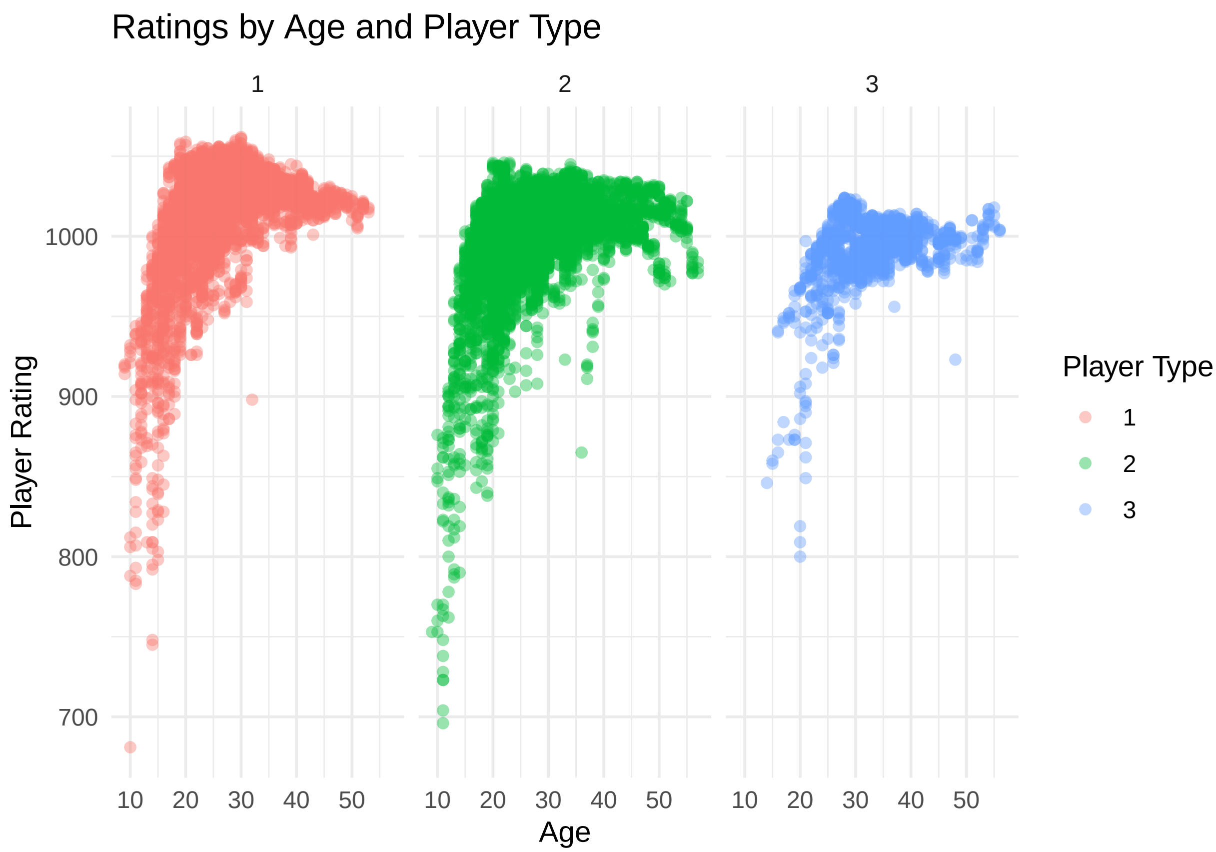

Figure 5: The cluster types shown as seperate graphs

```r
#performance by cluster on bar plots (MSE measure)
mean_SS_t_mod$Cluster <- colnames(cbind(clss, clss_sc, clss_hc, clss_kmeans))
ggplot(mean_SS_t_mod, aes(x = reorder(Cluster, Mean_MSE_PlayerType), y = Mean_MSE_PlayerTy
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "RMSE of Player-Type Models by Clustering Method",
       x = "Clustering Variable", y = "RMSE (Player-Type)") +
  theme_minimal()
```

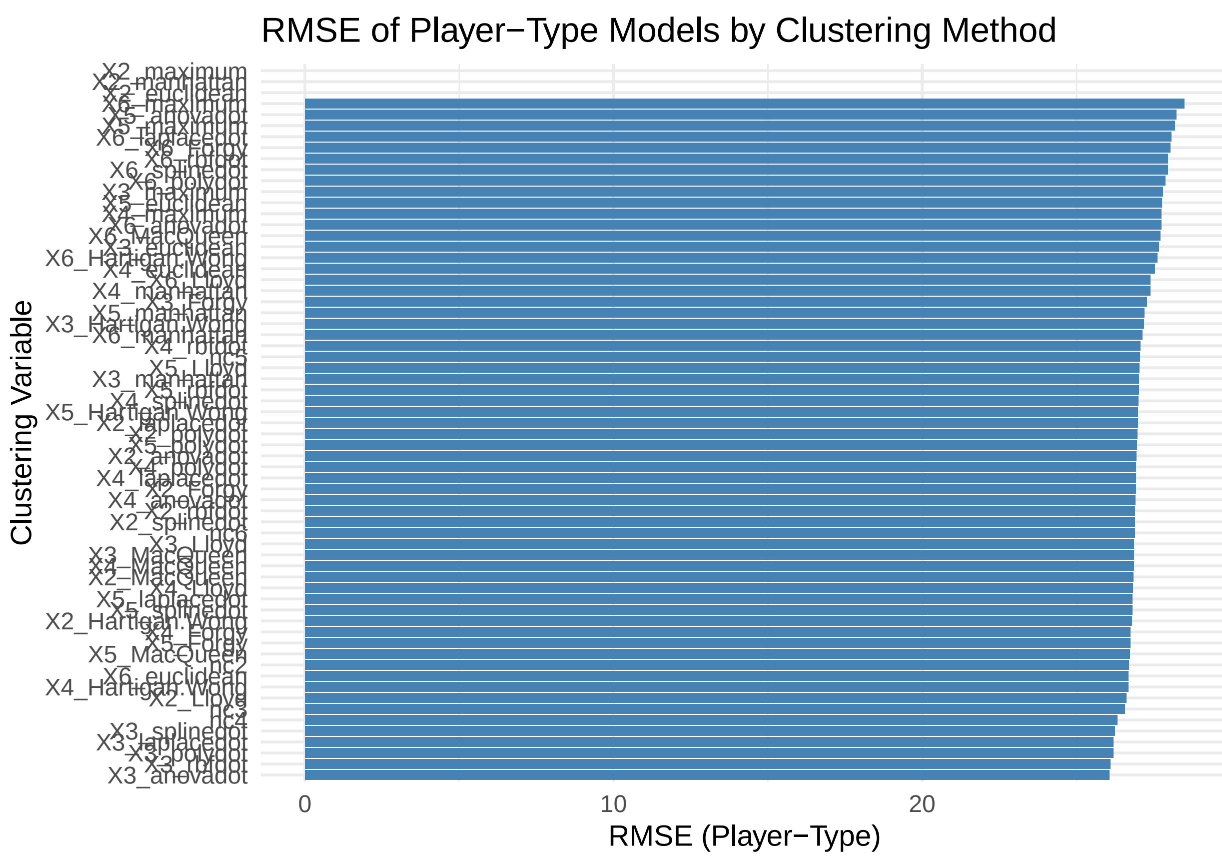

Figure 6: Bar plot of all clustering types and their RSME

Spline

The use of splines were included based off the fact that sometimes quadratic models sometimes over simplify the relationship between the ages and the ratings of the players. Spline regression gives more flexibility by enabling the model to capture more complexity and non linear trends in the data. It is also useful for when the effect of age on performance isn't constant across all different age ranges. The spline model for player type is as follows

$outp < -lm(pt_ratr\ pt_type * bs(pt_ages, degree = 3, df = df_try), data = MPO_ar_pt_train2)$

and the non player type

$out < -lm(ratr\ bs(ages, degree = 3, df = df_try), data = MPO_ar_pt_train)$

Something that differed from the quadratic model is the "DF_Used" on the last column. Each spline was fitted from 3:8 knots and then the best one was determined that yielded the best measurements of the model, like MSE, SSE, ect. The results are then all stored in a table.

Visualizations

The following visualizations are calculating and showing what clustering method was the best performance wise.

```r
# Best player type
names(MPO_ar_pt)[which.min(mean_SS_t$Mean_MSE_PlayerType) + 12]
```

```
## [1] "X6_manhattan"
```

```r
best_order <- names(MPO_ar_pt)[order(mean_SS_t$Mean_MSE_PlayerType) + 12]

mean_SS_t[order(mean_SS_t$Mean_MSE_PlayerType), ]
```

```
##    Mean_SSE_PlayerType Mean_SSE_NoPlayerType Mean_MSE_NoPlayerType
## 45           151634879             167886247              1295.709
## 35           151543573             167717151              1295.057
## 43           154754771             173091893              1315.644
## 7            153597438             172036018              1311.625
## 12           154574308             169956208              1303.673
## 22           155760516             176293344              1327.755
## 56           158847162             175064370              1323.119
## 17           155055795             179119558              1338.356
## 57           155658467             175268321              1323.889
## 44           155990290             176324245              1327.871
## 27           156170854             173939591              1318.862
## 42           155161973             173946511              1318.888
## 26           157174014             169582583              1302.239
## 3            157139648             173881157              1318.640
```

```
## 16    159085442    171935334    1311.241
## 28    157443841    172474245    1313.295
## 53    159328056    173561765    1317.428
## 21    159043820    175091835    1323.223
## 14    158095848    170599114    1306.136
## 1     160373720    175917843    1326.340
## 61    160329479    173628389    1317.681
## 51    157375224    173660348    1317.803
## 19    157795797    172366893    1312.886
## 52    161234095    174542721    1321.146
## 46    162042059    178522353    1336.123
## 58    160402258    171764970    1310.591
## 63    160530748    171556817    1309.797
## 49    159592932    174657118    1321.579
## 6     161503793    177799671    1333.415
## 8     162799258    180989406    1345.323
## 13    162641135    173170142    1315.941
## 11    160884376    172051922    1311.686
## 62    162529396    172974445    1315.197
## 29    163673196    175189865    1323.593
## 48    161516524    173628664    1317.682
## 9     163595603    169818842    1303.146
## 59    164894258    170857680    1307.125
## 54    167082632    176197144    1327.393
## 5     165900491    169173203    1300.666
## 18    165566239    184976249    1360.060
## 47    166636746    173038482    1315.441
## 64    166127527    175458512    1324.608
## 37    169168551    169134831    1300.518
## 23    169549196    172907697    1314.944
## 4     171051106    172747358    1314.334
## 33    169435460    169314574    1301.209
## 38    173654658    173616137    1317.635
## 32    173429303    173397141    1316.803
## 34    173491562    173477715    1317.109
## 25    177992505    171162711    1308.292
## 20    184963999    173045656    1315.468
```

```
## 50          182714866             176196848           1327.392
## 15          189101651             172249669           1312.439
## 24          186725493             173899437           1318.709
## 40          198127556             184307082           1357.597
## 10          198353738             183009190           1352.809
## 65          203800290             184631907           1358.793
## 60          202789652             185706135           1362.740
## 2           207166305             189696406           1377.303
## 55          212749065             176985324           1330.358
## 30          225678806             182473232           1350.827
## 39          290023349             171616596           1310.025
## 31                 NA                    NA                 NA
## 36                 NA                    NA                 NA
## 41                 NA                    NA                 NA
##    Mean_MSE_PlayerType Adj_R_Sqr_NoPlayerType Adj_R_Sqr_PlayerType DF_Used
## 45            24.01136              0.4111357            0.5138408       4
## 35            24.06100              0.4089544            0.4999376       4
## 43            24.16739              0.4114851            0.5028604       4
## 7             24.19220              0.4107794            0.5324238       4
## 12            24.20906              0.4088278            0.5324395       4
## 22            24.26094              0.4111664            0.5312982       4
## 56            24.29320              0.4081767            0.4827640       7
## 17            24.29965              0.4119891            0.4981729       3
## 57            24.30718              0.4141631            0.5305570       6
## 44            24.31781              0.4105548            0.5161999       7
## 27            24.31782              0.4110106            0.5333118       4
## 42            24.32967              0.4141214            0.5120357       7
## 26            24.37225              0.4099852            0.4776506       7
## 3             24.37382              0.4121511            0.5243097       4
## 16            24.42008              0.4135898            0.4802012       7
## 28            24.51959              0.4143897            0.5282713       4
## 53            24.52253              0.4100196            0.5027150       4
## 21            24.52926              0.4103566            0.4767276       7
## 14            24.57692              0.4069409            0.5267190       4
## 1             24.59793              0.4107465            0.4875872       8
## 61            24.64363              0.4142160            0.4824512       6
## 51            24.65287              0.4063466            0.4841900       7
```

```
## 19        24.65445       0.4060060       0.5302575     4
## 52        24.65574       0.4118402       0.5220482     4
## 46        24.65901       0.4099522       0.4801084     7
## 58        24.72049       0.4081770       0.5034039     5
## 63        24.73831       0.4103615       0.5026139     4
## 49        24.74864       0.4102072       0.5275040     4
## 6         24.76268       0.4157332       0.4839112     8
## 8         24.77863       0.4105767       0.4974694     3
## 13        24.86098       0.4141691       0.5280348     4
## 11        24.86539       0.4128709       0.4738649     4
## 62        24.87150       0.4104620       0.5310174     6
## 29        24.90459       0.4145668       0.5326670     4
## 48        24.91563       0.4125619       0.5038796     5
## 9         24.94147       0.4114992       0.5378008     6
## 59        25.07053       0.4156640       0.5249737     4
## 54        25.13180       0.4074494       0.5247016     4
## 5         25.16777       0.4096908       0.5240381     4
## 18        25.18225       0.4105913       0.4980751     3
## 47        25.25587       0.4091991       0.5313101     6
## 64        25.27613       0.4129712       0.5332200     4
## 37        25.28964       0.4117122       0.4116647     4
## 23        25.33111       0.4129367       0.5267817     4
## 4         25.42023       0.4156819       0.5183685     4
## 33        25.50468       0.4108346       0.4184841     4
## 38        25.59383       0.4106114       0.4125612     6
## 32        25.63162       0.4122443       0.4128760     4
## 34        25.64207       0.4097023       0.4201033     8
## 25        25.99625       0.4155896       0.5026153     4
## 20        26.52840       0.4102234       0.5087914     4
## 50        26.55635       0.4116925       0.5347702     4
## 15        26.91415       0.4063223       0.5000774     4
## 24        26.98060       0.4104939       0.4558698     4
## 40        27.40244       0.4128970       0.3946780     3
## 10        27.43750       0.4096671       0.4713444     3
## 65        27.77422       0.4114170       0.5068488     3
## 60        27.81056       0.4121508       0.5049772     3
## 2         27.91083       0.4152789       0.4753885     3
```

```
## 55          28.56006          0.4112929          0.4975350     3
## 30          29.20990          0.4089767          0.4731756     3
## 39          33.78178          0.4112985          0.4278301     5
## 31                NA                 NA                 NA    NA
## 36                NA                 NA                 NA    NA
## 41                NA                 NA                 NA    NA
```

```r
## table of best clustering types
tab_best_order <- data.frame(
  Best_cluster_types = c(best_order[1:6])
)

kable(tab_best_order)
```

Table 2: Table of top 6 best clusterting methods from splines

Best_cluster_types
X6_manhattan
X6_euclidean
X4_manhattan
X3_rbfdot
X3_polydot
X3_anovadot

```r
# Visualizing results
pp <- apply(clustdm_cv[, c(2, 3, 5, 6)], 2, function(x) tapply(x, clustdm_cv$X6_manhattan
barplot(t(pp), beside = TRUE, ylab = "Standardized Statistic (C1,C2, C1P, C2P)", xlab = "F
```

```r
# Final polynomial-based visualization for sanity check
out <- lm(ratr ~ -1 + factor(X6_manhattan) + factor(X6_manhattan) / ages + factor(X6_manh
          data = MPO_ar_pt)
summary(out)
```

```
##
## Call:
```

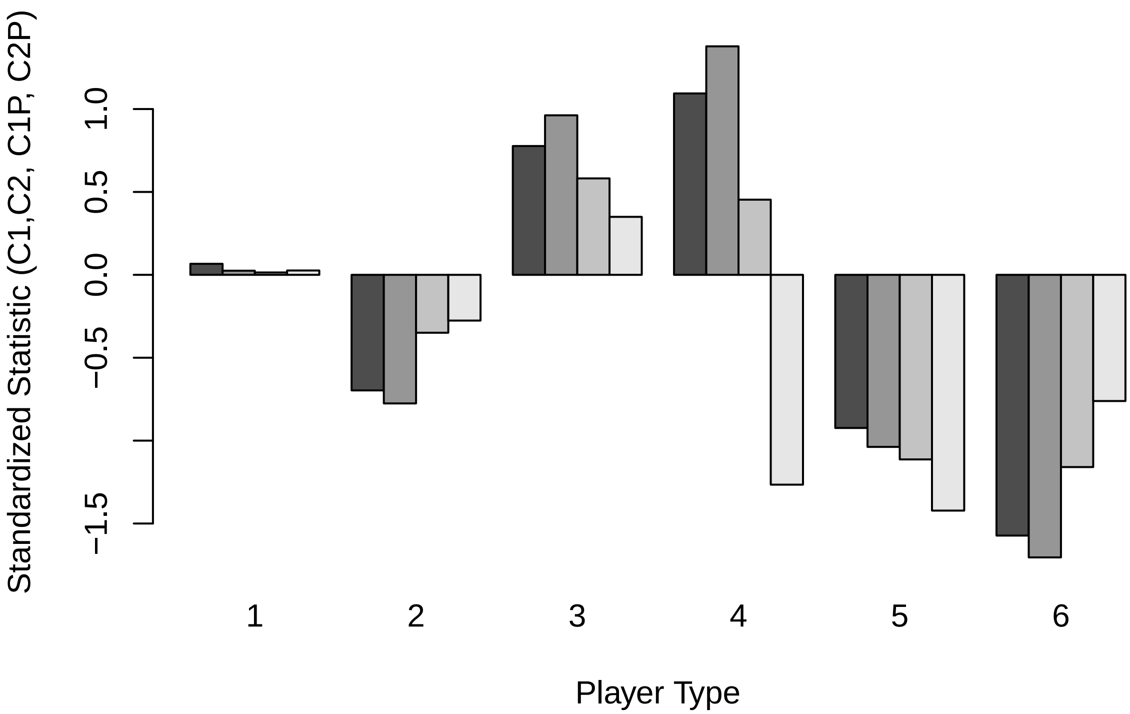

Figure 7: Bar plot of player data aross the different clusters from splines

```
## lm(formula = ratr ~ -1 + factor(X6_manhattan) + factor(X6_manhattan)/ages +
##     factor(X6_manhattan)/I(ages^2), data = MPO_ar_pt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -238.406  -11.277    1.649   14.749   66.347
##
## Coefficients:
##                                   Estimate Std. Error  t value Pr(>|t|)
## factor(X6_manhattan)1            1015.1278     0.3309 3068.087  < 2e-16 ***
## factor(X6_manhattan)2             988.5610     3.0049  328.978  < 2e-16 ***
## factor(X6_manhattan)3            1043.8113     0.6584 1585.323  < 2e-16 ***
## factor(X6_manhattan)4             897.7840    56.0691   16.012  < 2e-16 ***
## factor(X6_manhattan)1:ages         15.3001     0.3655   41.863  < 2e-16 ***
## factor(X6_manhattan)2:ages         35.3738     6.0854    5.813 6.28e-09 ***
## factor(X6_manhattan)3:ages          9.0485     0.7635   11.852  < 2e-16 ***
## factor(X6_manhattan)4:ages       -280.4051   113.1466   -2.478  0.01322 *
## factor(X6_manhattan)1:I(ages^2)   -17.5540     0.3618  -48.523  < 2e-16 ***
## factor(X6_manhattan)2:I(ages^2)   -37.3595     7.7788   -4.803 1.58e-06 ***
## factor(X6_manhattan)3:I(ages^2)   -28.7874     0.6602  -43.601  < 2e-16 ***
## factor(X6_manhattan)4:I(ages^2) -166.0349    54.0128   -3.074  0.00212 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.18 on 13282 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9994
## F-statistic: 1.764e+06 on 12 and 13282 DF,  p-value: < 2.2e-16
```

```r
ages_plot <- seq(-1, 1, 0.1)
ages_plot2 <- ages_plot * sd(MPO_ar_pt$Age, na.rm = TRUE) + mean(MPO_ar_pt$Age, na.rm = TF

rat_pred1 <- coef(out)[1] + coef(out)[5] * ages_plot + coef(out)[9]  * ages_plot^2
rat_pred2 <- coef(out)[2] + coef(out)[6] * ages_plot + coef(out)[10] * ages_plot^2
rat_pred3 <- coef(out)[3] + coef(out)[7] * ages_plot + coef(out)[11] * ages_plot^2
rat_pred4 <- coef(out)[4] + coef(out)[8] * ages_plot + coef(out)[12] * ages_plot^2

plot(rat_pred1 ~ ages_plot2, type = "l", lwd = 2, col = "black",
```

```r
     ylim = c(min(rat_pred1, rat_pred2, rat_pred3, rat_pred4),
              max(rat_pred1, rat_pred2, rat_pred3, rat_pred4)),
     ylab = "Predicted Rating", xlab = "Age",
     main = "Polynomial Aging Curves by Player Type (X6_manhattan)")
lines(rat_pred2 ~ ages_plot2, col = "blue", lwd = 2)
lines(rat_pred3 ~ ages_plot2, col = "red", lwd = 2)
lines(rat_pred4 ~ ages_plot2, col = "darkgreen", lwd = 2)
```

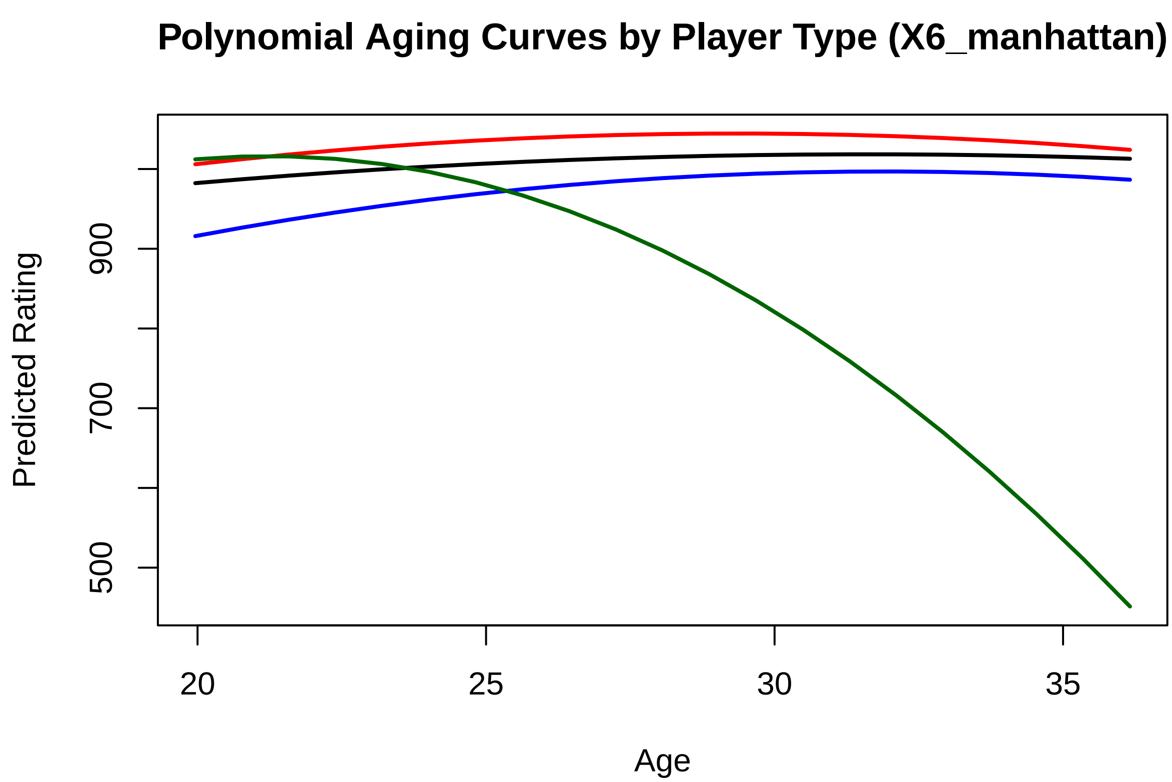

Figure 8: Polynomials from the different clusters from splines

These are a couple visualizations of of scatter plots of all the players.

```r
## scatter plot with different color points to show
ggplot(MPO_ar_pt, aes(x = Age, y = ratr)) +
  geom_point(alpha = 0.3) +
  labs(title = "All Players' Ratings by Age",
       x = "Age", y = "Player Rating") +
  theme_minimal()
```

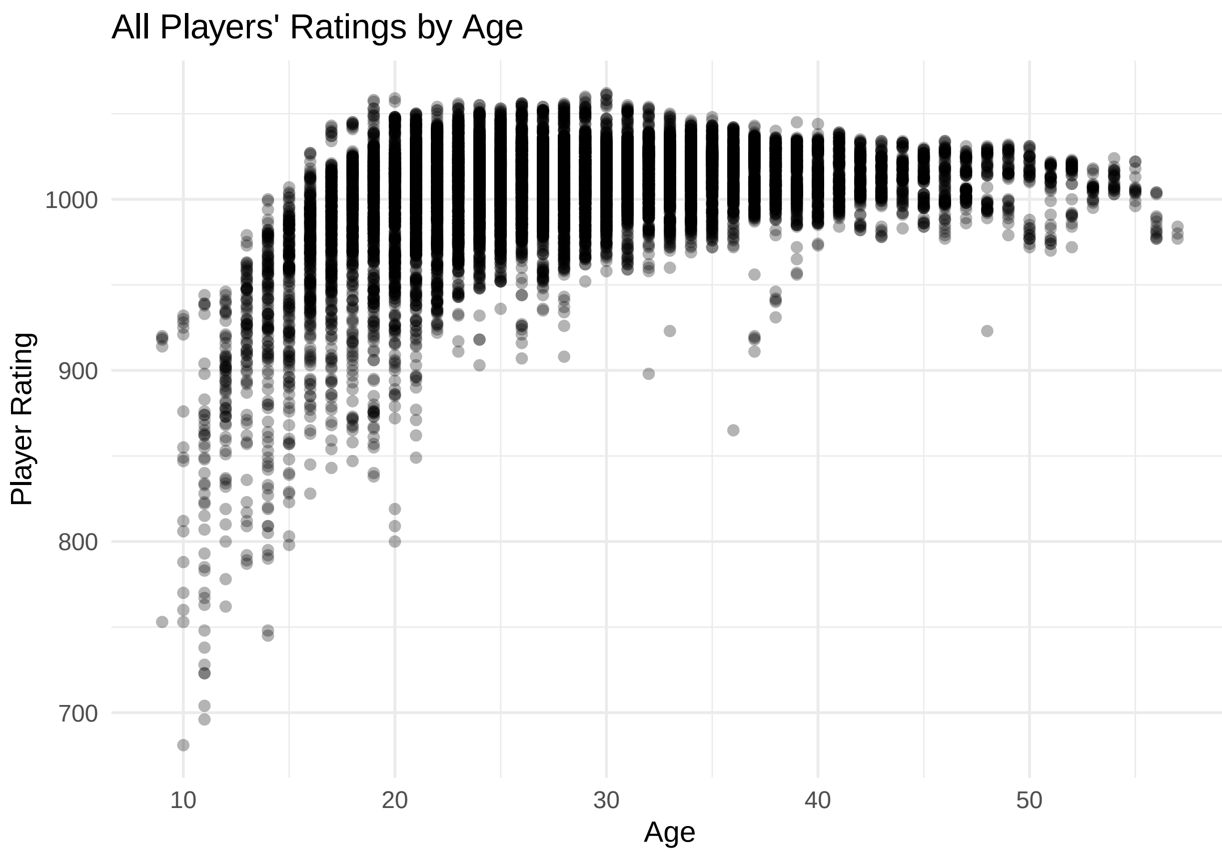

Figure 9: Scatter plot of all players

```r
# with color
ggplot(MPO_ar_pt, aes(x = Age, y = ratr, color = as.factor(`X6_manhattan`))) +
  geom_point(alpha = 0.4) +
  labs(title = "Ratings by Age and Player Type",
       x = "Age", y = "Player Rating", color = "Player Type") +
  theme_minimal()
```

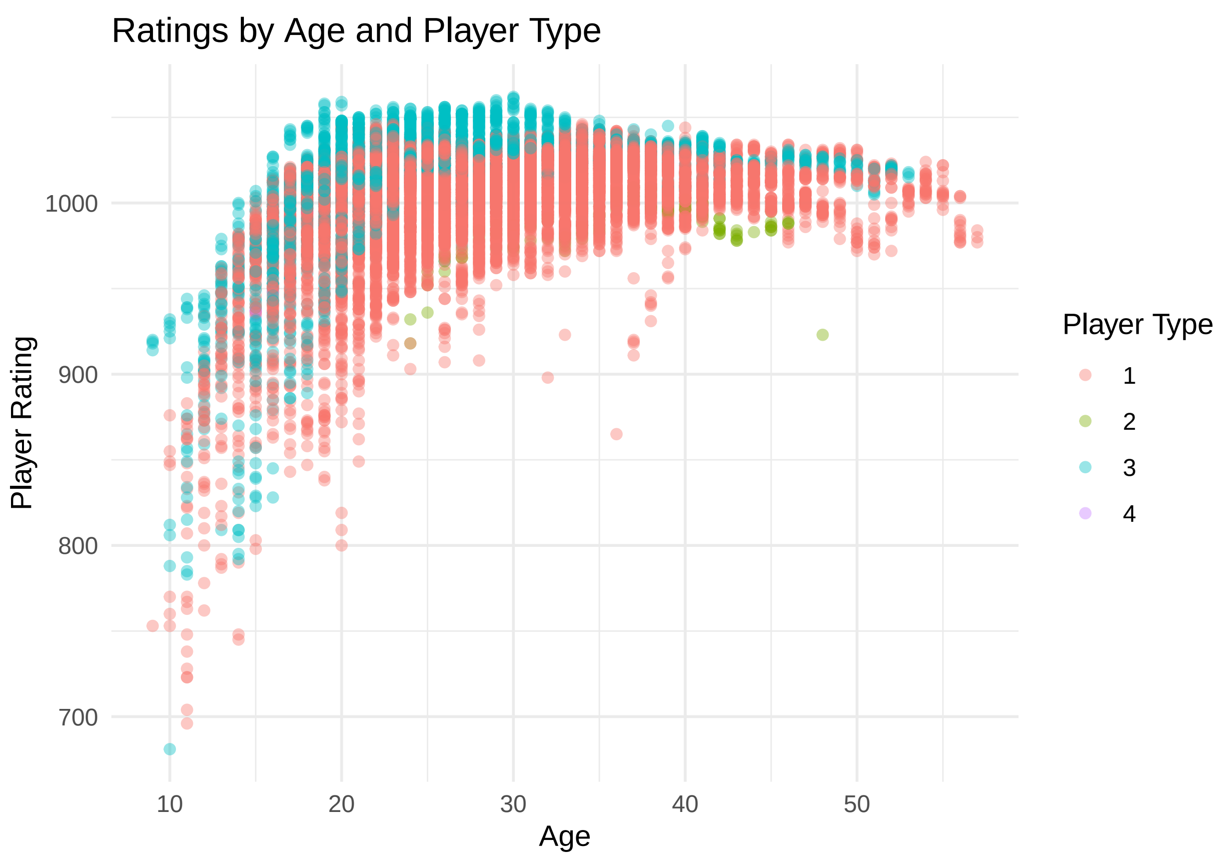

Figure 10: Seperated by color using clustering method

```r
# using facet wrap
ggplot(MPO_ar_pt, aes(x = Age, y = ratr, color = as.factor(`X6_manhattan`))) +
  geom_point(alpha = 0.4) +
  facet_wrap(~`X6_manhattan`) +
  labs(title = "Ratings by Age and Player Type",
       x = "Age", y = "Player Rating", color = "Player Type") +
  theme_minimal()
```

This is a representation of all the different clustering methods used and their effectiveness.

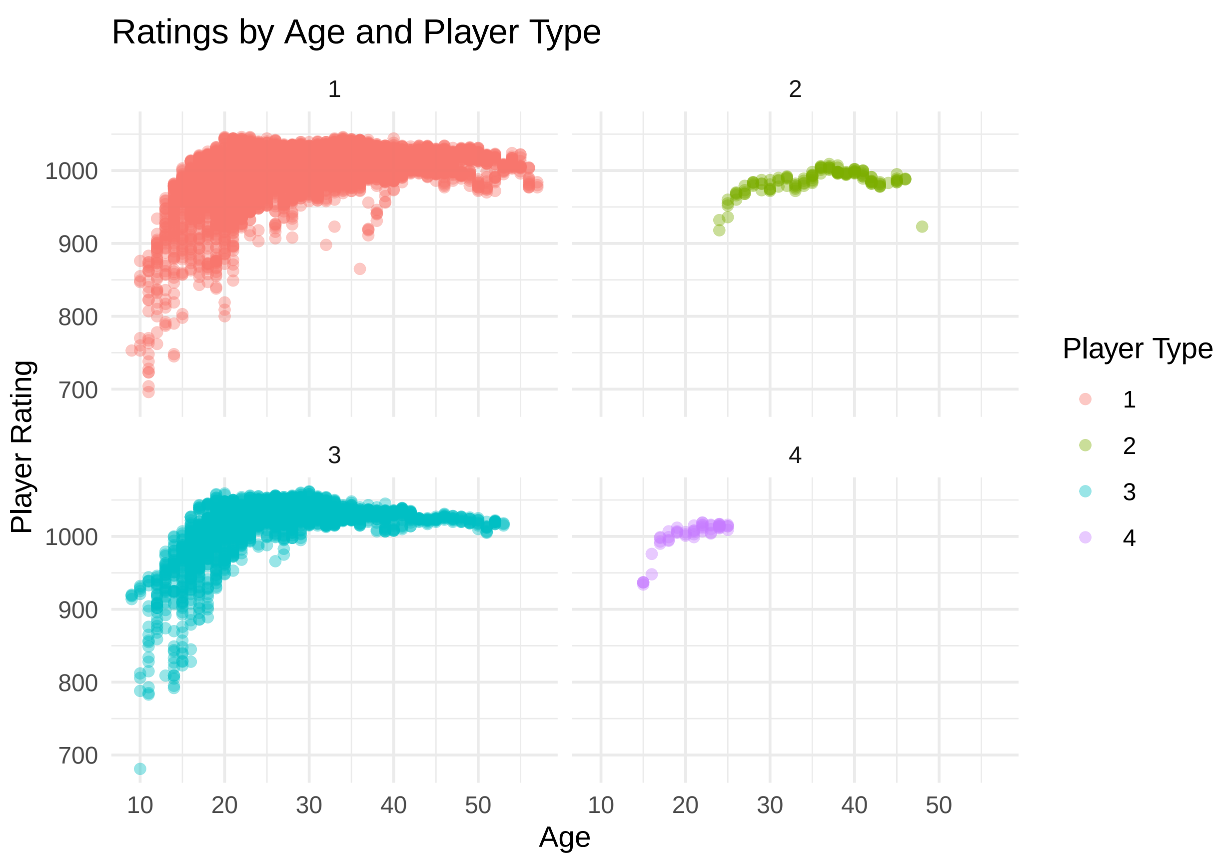

Figure 11: Different groups shown as individual graphs

```r
#performance by cluster on bar plots (MSE measure)
mean_SS_t$Cluster <- colnames(cbind(clss, clss_sc, clss_hc, clss_kmeans))
ggplot(mean_SS_t, aes(x = reorder(Cluster, Mean_MSE_PlayerType), y = Mean_MSE_PlayerType)
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "RMSE of Player-Type Models by Clustering Method",
       x = "Clustering Variable", y = "RMSE (Player-Type)") +
  theme_minimal()
```

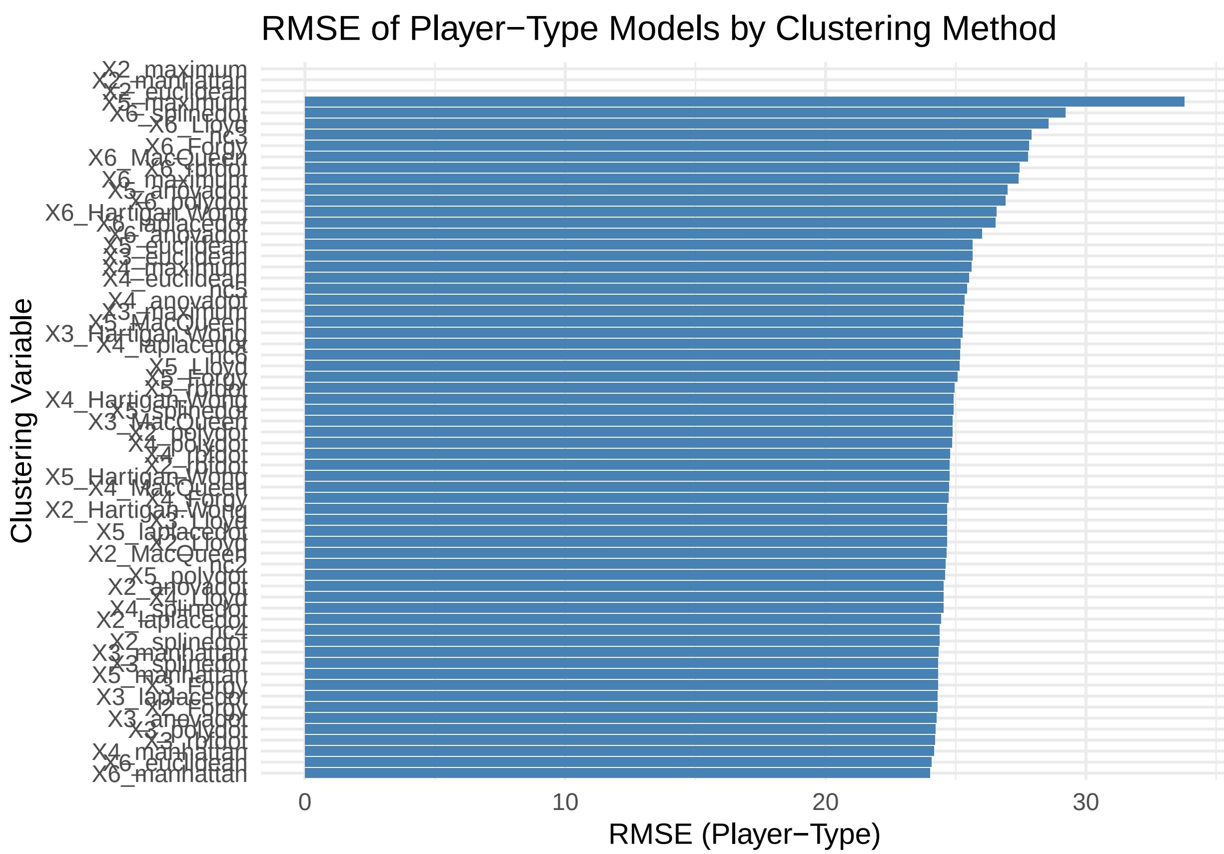

Figure 12: Bar plot of all clustering types and their RMSE

Discussion

Looking at our results holistically is how we can interpret the full scope of our research. When looking at the table of results for the quadratic model, We can interpret the performance of the player type versus the non player type. When solely looking at the MSE's of both, we can see that the player type model performed slightly better more often than the non player type. The player type (pt) had an MSE of around 26-27 every time, while the non player type (npt) had 27 or above. In retrospect we would still want the MSE in general to be lower, as on average this means that it was missing the true rating by around 26 points every time regardless.

In terms of the spline models, it had different performance. The MSE was around 24-27 for the pt, while the npt was all over the place, being in the 1300s. Based off that it would be a safe conclusion to say that when using splines for aging curves, it makes more sense to use pt models rather than not.

When looking at the bar plot of the clusters and the line plot of the aging curve of different clusters, there is clear differences that are drawn between players. For the bar plot the first group is the super stars, the second is the middle of the way players, and the third are the fringe lesser players. We can then see exactly why the players were split up in the way that they were. As opposed to the ones of the spline model, it is harder to interpret the six different groups that everything was split up by. The bar graph follows the quadratic model mostly, having good, middle of the way, and bad players grouped together. But then with cluster 4 it has a large mix of both, which is questionable. As well as with the aging curve plot, that shows the 4 splines that was fit. It seems as if it has 3 very closely contested groups, and then a very poor quickly decaying group.

The scatter plots that are a representation of all players give us a good insight as well. It shows the color corresponding to the group that the player belongs to. The quadratic model gives us a nicer representation of everything encompassing compared to the the spline model. You can clearly see where the difference between the groups of players are and how the aging curve line from our plots before exists.

Despite the extent of the research , some limitations and inconsistencies are present. Player ratings are influenced by many different external factors that were not captured in this data. Further more, players may transition their play style and skill patterns as they age, as they improve, and as the game changes. Some of the results from our spline models lacked aspects of interpreting, which may allude to over fitting. As far as real world context, research like this can be used by coaches, sponsors, and scouts and analysts to find high profile players easier. This could also help chart a players decline over their career and help guide their career as it declines.

Conclusion

This project set out to compare the effectiveness of modeling the aging curves through the sport of disc golf and their ratings of players. By fittitng both a quadratic model and a spline model across multiple different clustering strategies, we found that player type models edged out non player type model in terms of predictive error. The results suggest that athlete aging patterns are meaningful across different aging groups.

There is a strong potential for this type of modeling to be ued in other sports. It would be better utilized for individual performance based sports, like tennis, golf, and e sports. The approach of using clustering to define roles, followed by role specific modeling, could be space for expansion as well.

If this research was to be continued the first avenue that would be explored is the change in rating over time. Use of something like this would allow more dynamic modeling rating as a function of age alone. It might uncover transition patterns between roles when a player's performances changes because of injury, age, or improvement.