Exploratory Data Analysis

Dutchak Bohdan

This is the EDA for my project. The idea is to classify the position of the player given parformance stats of his average game. I have collected this data from the NBA Reference (https://www.basketball-reference.com/). This data contains ~10,000 individual players stats per game for seasons 2000-2020. It is collected by my data_parser.py (https://github.com/bohdan-dutchak/NBAdrafter/blob/main/data_parser.py) script

```
#install.packages("devtools")
#install.packages("psych")
#install.packages("dplyr")
#install.packages("ggcorrplot")
#install.packages("ggplot2")
#install.packages("plotly")
#install.packages("gridExtra")
#install.packages("ggpubr")
#install.packages("reshape2")
#install.packages("GGally")
#install.packages("broom")
```

```
library(devtools)
library(psych)
library(dplyr)
library(ggcorrplot)
library(ggplot2)
library(plotly)
library(ggpubr)
library(reshape2)
library(GGally)
library(broom)
```

```
data = read.csv('data/all_seasons.csv', head=T, sep=',')
head(data)
```

	Player <chr>	Pos <chr>	Ht <int>	Wt <int>	Exp <int>	Age <int>	G <int>	GS <int>	MP <dbl>
1	Nick Anderson	SG	198	93	10	32	72	72	29.1
2	Jon Barry	SG	193	88	7	30	62	1	20.7
3	Tyrone Corbin	SF	198	95	14	37	54	5	17.4
4	Tony Delk	PG	185	86	3	26	46	1	14.8
5	Vlade Divac	C	216	110	10	31	82	81	29.0
6	Lawrence Funderburke	PF	206	104	2	29	75	1	13.7

6 rows | 1-10 of 33 columns

Data Exploration

Let’s get familiar with the data.

Features description

```
names(data)
```

```
##  [1] "Player" "Pos"    "Ht"     "Wt"     "Exp"    "Age"    "G"      "GS"
##  [9] "MP"     "FG"     "FGA"    "FG."    "X3P"    "X3PA"   "X3P."   "X2P"
## [17] "X2PA"   "X2P."   "eFG."   "FT"     "FTA"    "FT."    "ORB"    "DRB"
## [25] "TRB"    "AST"    "STL"    "BLK"    "TOV"    "PF"     "PTS.G"  "Season"
```

Name	Data type	Feature type	Measurement	Description
Player	string	categorical	none	Full player’ name
Pos	string	categorical (**target**)	5 unique classes	Players position in a team: Center, Power Forward, Small Forward, Shooting Guard, Point Guard
Ht	float	numerical	cm	Players height
Wt	float	numerical	kg	Players weight
Exp	int	numerical	years	Years in the NBA
Age	int	numerical	full years	Players age
G	int	numerical	none	Games played in the season
GS	int	numerical	none	Games started
MP	float	numerical	minutes	Minutes played per game
FG	float	numerical	field goals	Field goals per game
FGA	float	numerical	attemps	Field goal attempts per game
FG%	float	numerical	percentage	% of successful fied goals in the season
3P	float	numerical	shots	3-pointers per game
3PA	float	numerical	attemps	3-point goal attempts per game

Name	Data type	Feature type	Measurement	Description
3P%	float	numerical	percentage	% of successful 3-point goals in the season
2P	float	numerical	shots	2-pointers per game
2PA	float	numerical	attemps	2-point goal attempts per game
2P%	float	numerical	percentage	% of successful 2-point goals in the season
eFG%	float	numerical	percentage	Effective Field Goal Percentage
FT	float	numerical	shots	Free throws per game
FTA	float	numerical	attemps	Free throws attempts per game
FT%	float	numerical	percentage	% of successful Free throws in the season
ORB	float	numerical	rebounds	Offensive Rebounds Per Game
DRB	float	numerical	rebounds	Deffensive Rebounds Per Game
TRB	float	numerical	rebounds	Total Rebounds Per Game
AST	float	numerical	assists	Assists per game
STL	float	numerical	steals	Steals per game
BLK	float	numerical	blocks	blocks per game
TOV	float	numerical	blocks	Turnovers per game
PF	float	numerical	fouls	Personal fouls per game
PTS/G	float	numerical	points	Points per game
Season	int	categorical	year	season

Missing Data

First of all, let’s see whether we have any missing values. If so, what should we do with them

```
options(warn=-1)
colSums(is.na(data))
```

```
## Player    Pos     Ht     Wt    Exp    Age      G     GS     MP     FG    FGA
##      0      0      0      0      0      0      0      0      0      0      0
##    FG.    X3P   X3PA   X3P.    X2P   X2PA   X2P.   eFG.     FT    FTA    FT.
##     65      0      0   1743      0      0    113     65      0      0    557
##    ORB    DRB    TRB    AST    STL    BLK    TOV     PF  PTS.G Season
##      0      0      0      0      0      0      0      0      0      0
```

```
X2Pna <- data[is.na(data$X2P.),]
head(X2Pna[16:18])
```

	X2P <dbl>	X2PA <dbl>	X2P. <dbl>
36	0	0	*NA*
216	0	0	*NA*
416	0	0	*NA*
605	0	0	*NA*
705	0	0	*NA*
772	0	0	*NA*

6 rows

```
X3Pna <- data[is.na(data$X3P.),]
head(X3Pna[13:15])
```

	X3P <dbl>	X3PA <dbl>	X3P. <dbl>
8	0	0	*NA*
12	0	0	*NA*
14	0	0	*NA*
24	0	0	*NA*
25	0	0	*NA*
26	0	0	*NA*

6 rows

Missing values analysis shows, that NA type is only in the rows, where player took 0 attempts of some sort of shot. Since the table could not divide by zero to get his % of shots, we don't have any value there, so we can fill it manually with 0's.

```
data[is.na(data)] <- 0
```

Univariate Analysis

When exploring our dataset and its features, we have many options available to us. We can explore each feature individually, or compare pairs of features, finding the correlation between. Let's start with some simple Univariate (one feature) analysis.

```
describe(data)
```

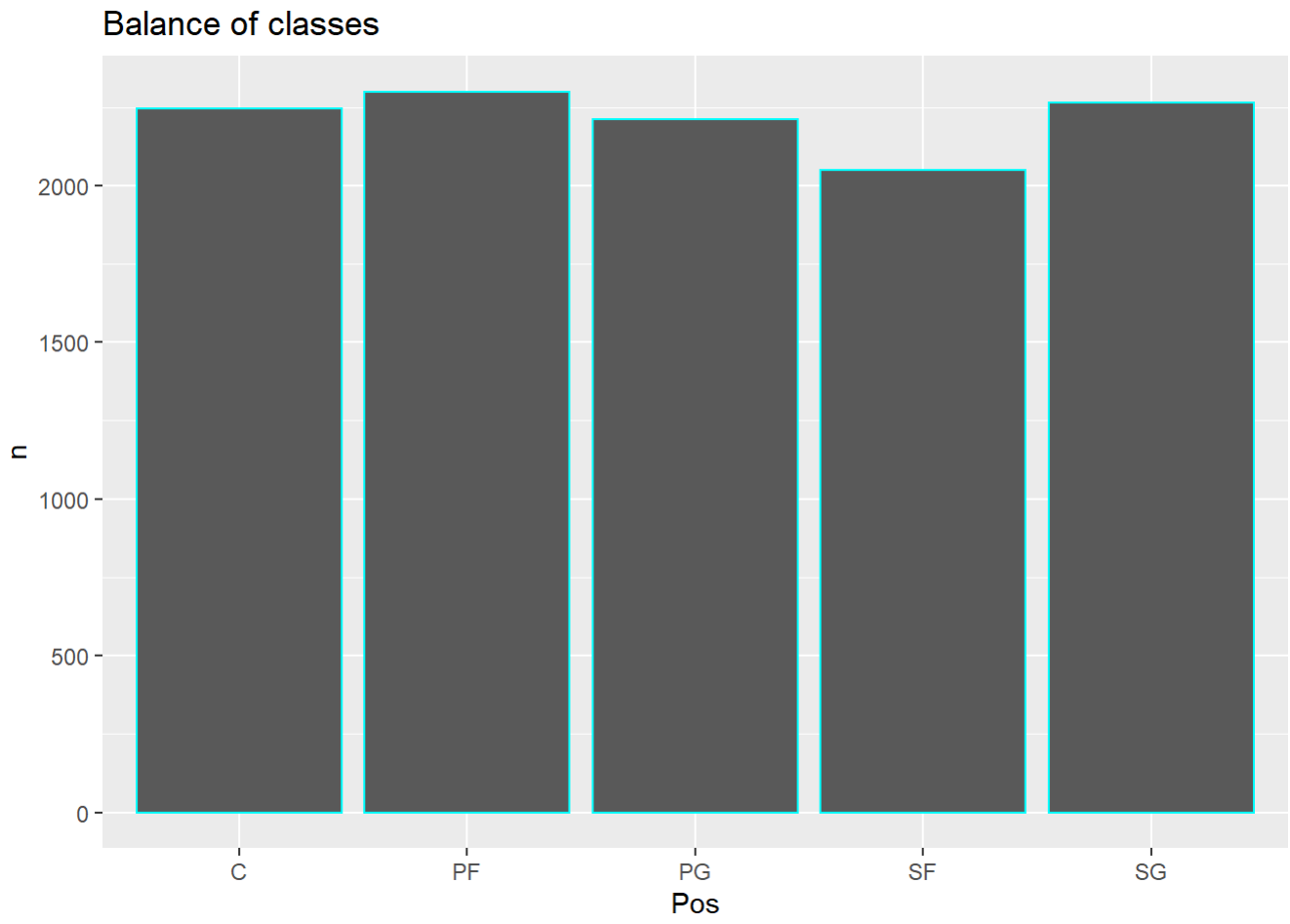
	v…	n	mean	sd	median	trimmed	mad	min	m
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<db
Player*	1	11071	1011.0335110	586.6800760	1011.000	1011.1261149	759.0912000	1	2033
Pos*	2	11071	2.9808509	1.4222373	3.000	2.9760641	1.4826000	1	5
Ht	3	11071	200.3395357	9.1927699	201.000	200.7694479	10.3782000	160	231
Wt	4	11071	99.8982928	12.4288816	100.000	99.6409620	13.3434000	61	163
Exp	5	11071	4.6453798	4.0736863	4.000	4.1928418	4.4478000	0	21
Age	6	11071	26.6915364	4.2937277	26.000	26.4176358	4.4478000	18	44
G	7	11071	46.9135579	26.5961505	51.000	47.9841933	35.5824000	1	82
GS	8	11071	22.7739138	28.0286360	8.000	18.5898160	11.8608000	0	82
MP	9	11071	19.9033963	10.0387861	19.200	19.7277295	12.1573200	0	43
FG	10	11071	2.9749887	2.1556473	2.500	2.7298182	2.0756400	0	12

1-10 of 32 rows | 1-10 of 14 columns Previous **1** 2 3 4 Next

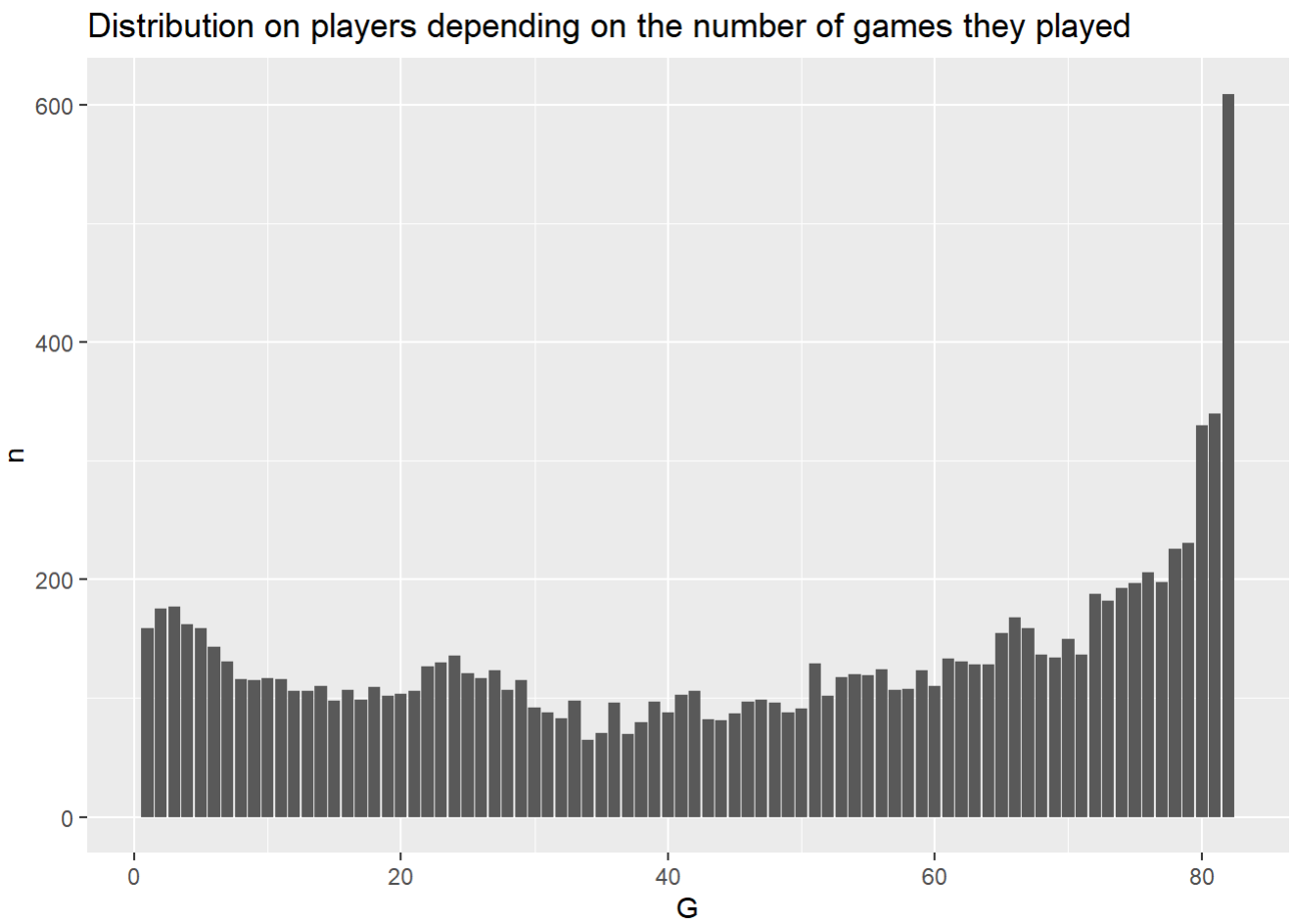
The dataset has 11071 observations and 32 features, 3 of them are categorical: Player, Pos and Season, all the other are numerical. Among the 11071 observations, only 2033 are unique. It means that the data is only about 2033 unique players and their performance in different seasons. I decided to collect seasons separately, since a player could change his position between the seasons.

We also need to know how good the data is balanced and what is it's distributions in order to make further decisions. The bar plot shows the distribution of classes in the target variable.

```
g <- data %>% count(Pos)
ggplot(g, aes(x=Pos, y=n)) + geom_bar(stat='identity', col="cyan") + ggtitle("Balance of classe
s")
```




```
g <- data %>% count(G)
ggplot(g, aes(x=G, y=n)) + geom_bar(stat='identity') + ggtitle("Distribution on players dependin
g on the number of games they played")
```



To decide what to do with “unrepresentative” observations, i.e., where the player participated only in a relatively small number of games, look at the histogram. Fortunately, there is a similar amount of such players and players who played the “middle” number of games. The bigger this number, the more accurate stats are. So I decided to keep such observations.

```
options(warn=-1)
t(aggregate(data, list(data$Pos), FUN=mean))[3:33,]
```

```
##        [,1]         [,2]         [,3]         [,4]         [,5]
## Pos    NA           NA           NA           NA           NA
## Ht     "210.8179"   "205.9835"   "187.5613"   "201.7632"   "195.4064"
## Wt     "113.84506"  "107.55043"  " 85.11126"  " 99.72314"  " 92.89409"
## Exp    "4.977738"   "4.805217"   "4.609227"   "4.532227"   "4.291262"
## Age    "26.99288"   "26.66696"   "26.85844"   "26.52344"   "26.40688"
## G      "46.66919"   "46.56826"   "46.45455"   "47.77734"   "47.17343"
## GS     "23.47240"   "21.54826"   "22.71551"   "24.26367"   "22.03619"
## MP     "17.70098"   "19.19548"   "20.78204"   "21.05820"   "20.90388"
## FG     "2.614782"   "2.959087"   "3.029444"   "3.085449"   "3.195190"
## FGA    "5.231879"   "6.403087"   "7.206649"   "7.081787"   "7.536981"
## FG.    "0.4818954"  "0.4446257"  "0.4023541"  "0.4183472"  "0.4053023"
## X3P    "0.09532502" "0.38573913" "0.76024423" "0.74458008" "0.91575463"
## X3PA   "0.2958148"  "1.1324783"  "2.2047942"  "2.1244141"  "2.5767432"
## X3P.   "0.09160508" "0.19292783" "0.30425373" "0.29784619" "0.31512048"
## X2P    "2.517809"   "2.573826"   "2.270149"   "2.341797"   "2.279921"
## X2PA   "4.934817"   "5.271130"   "5.003030"   "4.957373"   "4.961827"
## X2P.   "0.4912863"  "0.4699652"  "0.4326119"  "0.4552144"  "0.4407648"
## eFG.   "0.4887809"  "0.4721378"  "0.4528955"  "0.4710903"  "0.4655684"
## FT     "1.299466"   "1.395783"   "1.489959"   "1.463770"   "1.522374"
## FTA    "1.931790"   "1.934522"   "1.870782"   "1.912402"   "1.920521"
## FT.    "0.6253166"  "0.6594017"  "0.7349833"  "0.6973638"  "0.7298416"
## ORB    "1.5690116"  "1.3094783"  "0.4106287"  "0.7888672"  "0.5034422"
## DRB    "3.308816"   "3.190478"   "1.784487"   "2.526367"   "1.962798"
## TRB    "4.875913"   "4.497826"   "2.192944"   "3.313232"   "2.463283"
## AST    "0.8673642"  "1.1150000"  "3.5502035"  "1.4600098"  "1.8669462"
## STL    "0.4313001"  "0.5336522"  "0.7901854"  "0.6837402"  "0.6961606"
## BLK    "0.8002671"  "0.5035217"  "0.1374943"  "0.3277832"  "0.2080318"
## TOV    "1.008014"   "1.032478"   "1.484848"   "1.072021"   "1.145808"
## PF     "2.129029"   "1.948652"   "1.610493"   "1.732715"   "1.596778"
## PTS.G  "6.622039"   "7.696652"   "8.302668"   "8.377979"   "8.825905"
## Season "2010.152"   "2010.389"   "2010.249"   "2010.345"   "2010.959"
```

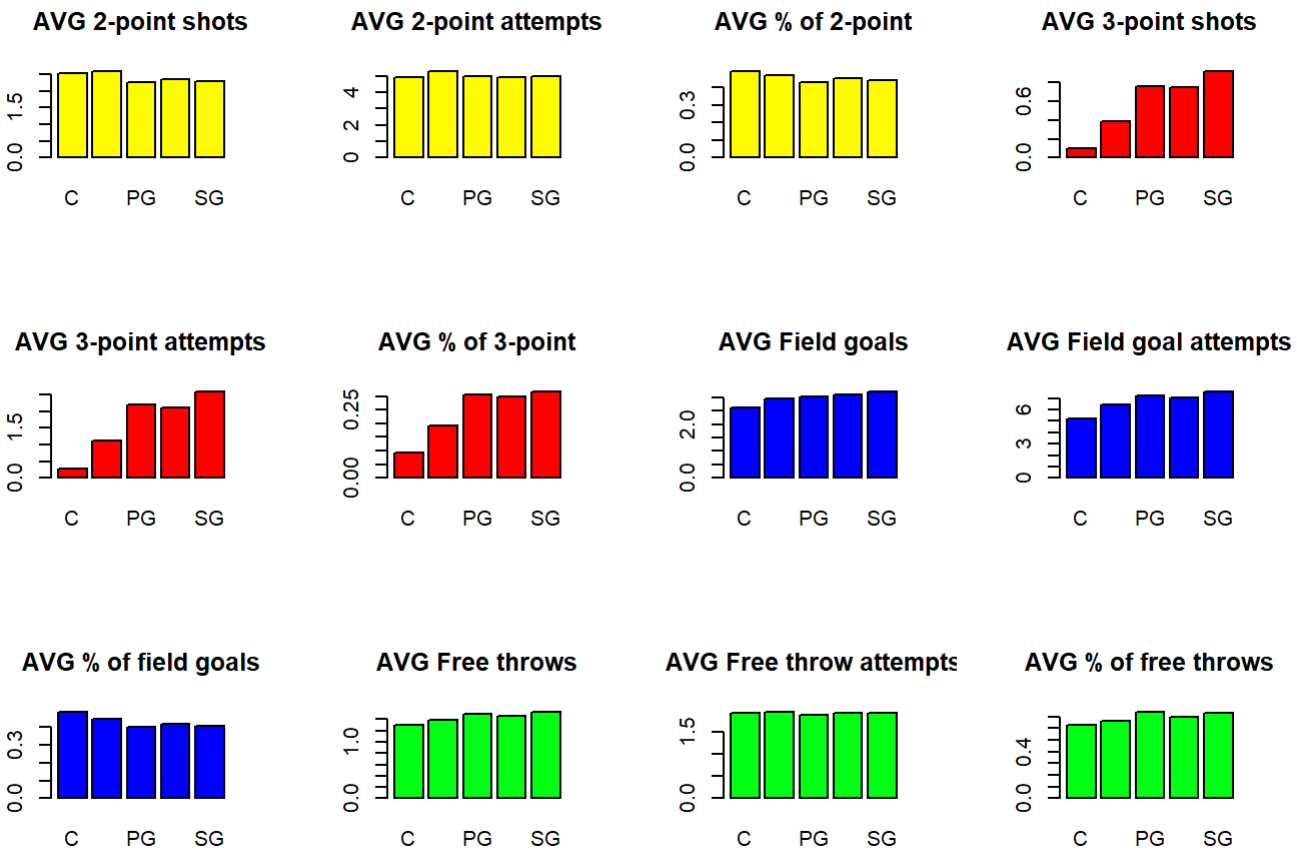
Taking a glance at the average values of each feature distributed by columns, we can notice some values age siglificantly different, but let's look at it in more representative way plotting it.

Bi-variate Analysis

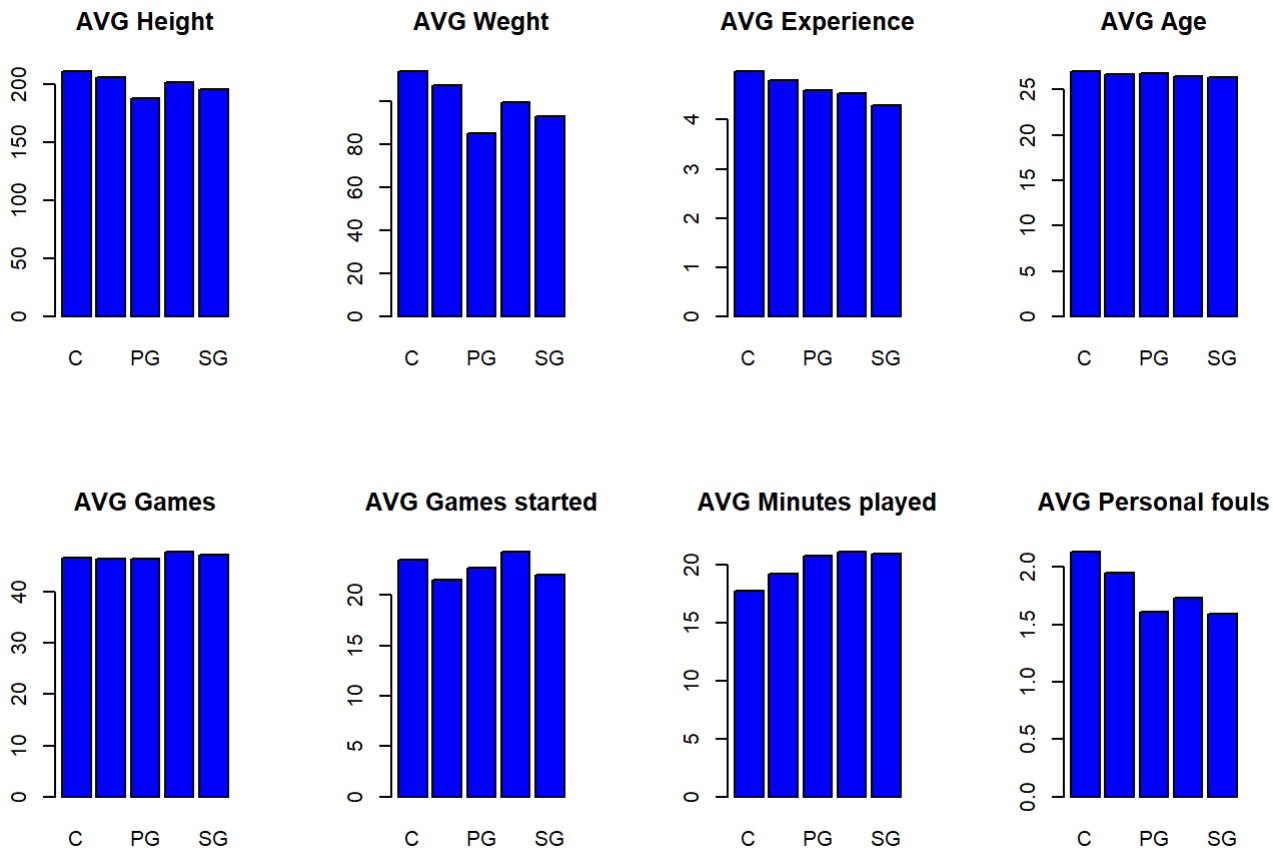
So far, we have analised all features individually. Let's now start combining some of these features together to obtain further insight into the interactions between them.

```
data_by_pos <- aggregate(data, list(data$Pos), mean)
attach(mtcars)
```

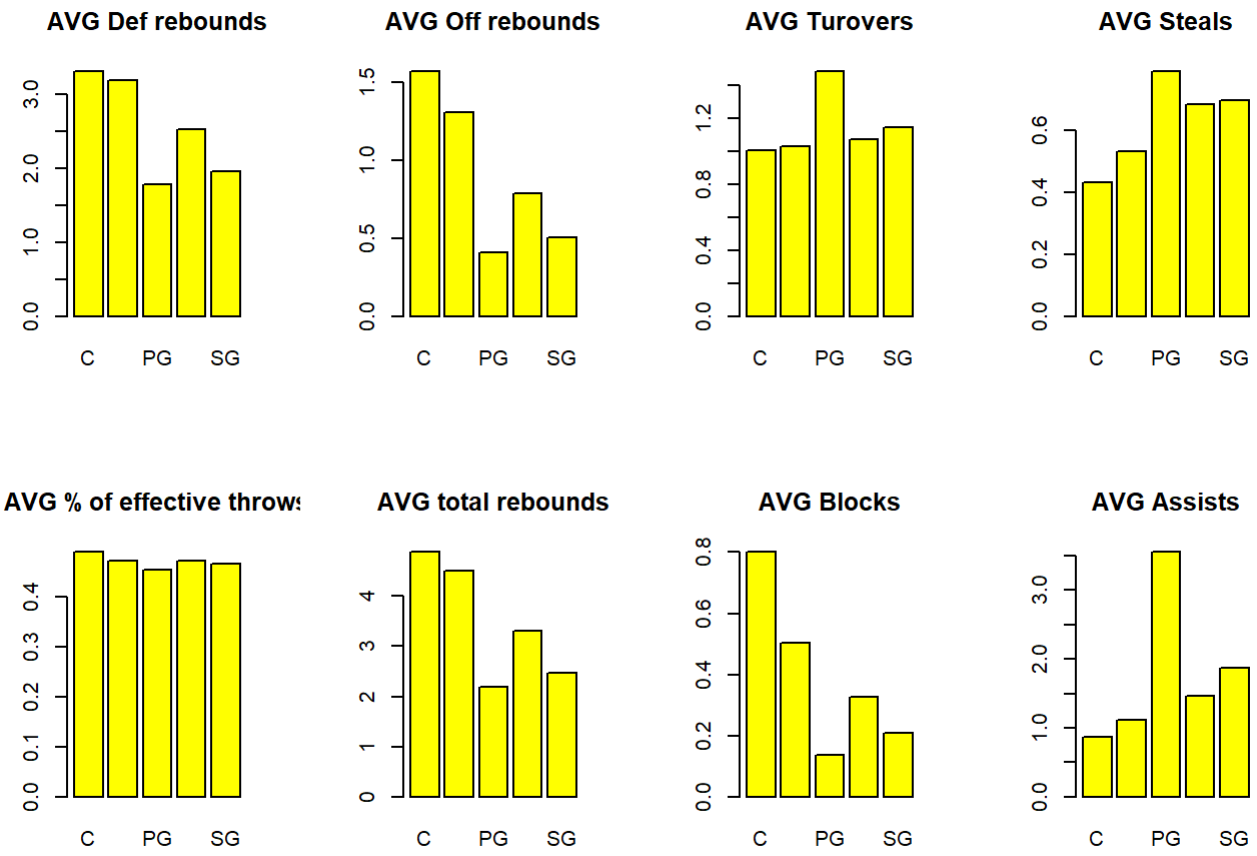
```
par(mfrow=c(3,4))
barplot(data_by_pos$X2P, main="AVG 2-point shots", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="#fffb00")
barplot(data_by_pos$X2PA, main="AVG 2-point attempts", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'
), col="#fffb00")
barplot(data_by_pos$X2P., main="AVG % of 2-point", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="#fffb00")
barplot(data_by_pos$X3P, main="AVG 3-point shots", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="red")
barplot(data_by_pos$X3PA, main="AVG 3-point attempts", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'
), col="red")
barplot(data_by_pos$X3P., main="AVG % of 3-point", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="red")

barplot(data_by_pos$FG, main="AVG Field goals", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col=
"blue")
barplot(data_by_pos$FGA, main="AVG Field goal attempts", names.arg = c('C', 'PF', 'PG', 'SF', 'S
G'), col="blue")
barplot(data_by_pos$FG., main="AVG % of field goals", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'
), col="blue")
barplot(data_by_pos$FT, main="AVG Free throws", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col=
"#00ff15")
barplot(data_by_pos$FTA, main="AVG Free throw attempts", names.arg = c('C', 'PF', 'PG', 'SF', 'S
G'), col="#00ff15")
barplot(data_by_pos$FT., main="AVG % of free throws", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'
), col="#00ff15")
```



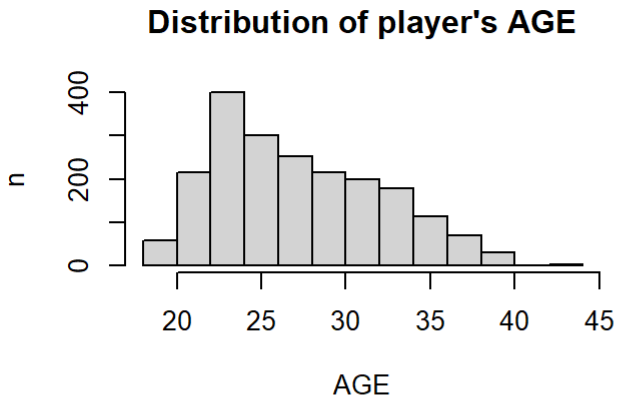

```
attach(mtcars)
par(mfrow=c(2,4))

barplot(data_by_pos$Ht, main="AVG Height", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="blu
e")
barplot(data_by_pos$Wt, main="AVG Weght", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="blue"
)
barplot(data_by_pos$Exp, main="AVG Experience", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col=
"blue")
barplot(data_by_pos$Age, main="AVG Age", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="blue")
barplot(data_by_pos$G, main="AVG Games", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="blue")
barplot(data_by_pos$GS, main="AVG Games started", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), co
l="blue")
barplot(data_by_pos$MP, main="AVG Minutes played", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="blue")
barplot(data_by_pos$PF, main="AVG Personal fouls", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), c
ol="blue")
```

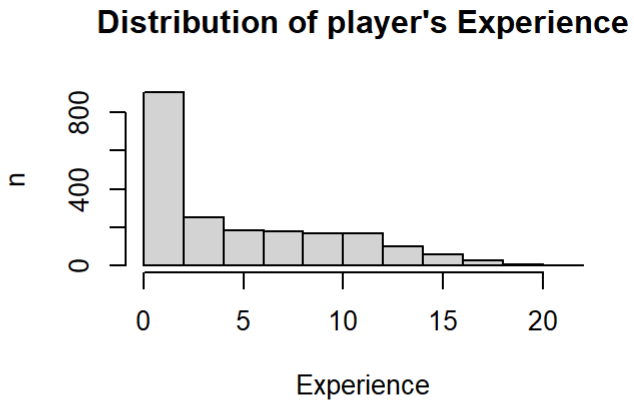


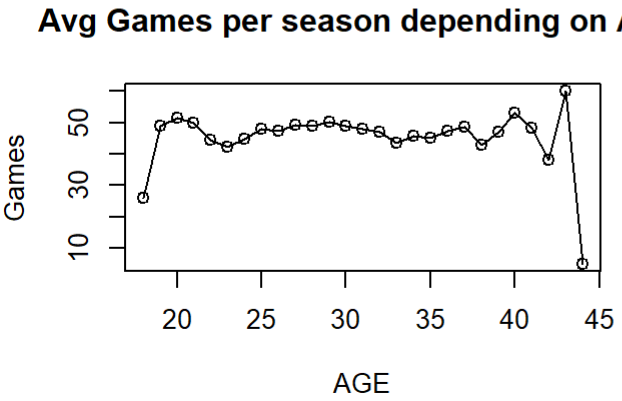

```
attach(mtcars)
par(mfrow=c(2,4))
barplot(data_by_pos$DRB, main="AVG Def rebounds", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), co
l="yellow")
barplot(data_by_pos$ORB, main="AVG Off rebounds", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), co
l="yellow")
barplot(data_by_pos$TOV, main="AVG Turovers", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="y
ellow")
barplot(data_by_pos$STL, main="AVG Steals", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="yel
low")
barplot(data_by_pos$eFG., main="AVG % of effective throws", names.arg = c('C', 'PF', 'PG', 'SF',
'SG'), col="yellow")
barplot(data_by_pos$TRB, main="AVG total rebounds", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'),
 col="yellow")
barplot(data_by_pos$BLK, main="AVG Blocks", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="yel
low")
barplot(data_by_pos$AST, main="AVG Assists", names.arg = c('C', 'PF', 'PG', 'SF', 'SG'), col="ye
llow")
```

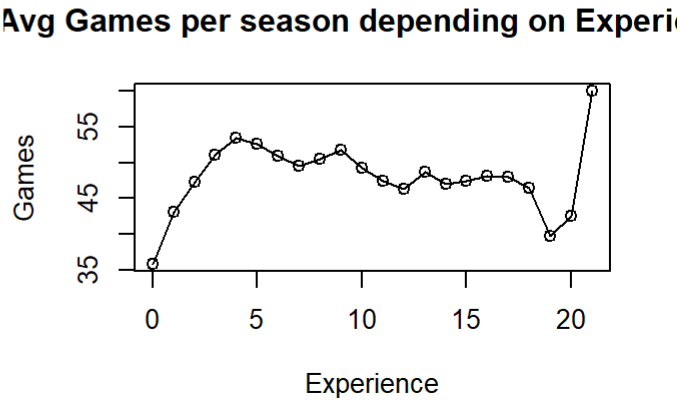



```
attach(mtcars)

par(mfrow=c(2,2))

hist(aggregate(data, list(data$Player), max)$Age, main="Distribution of player's AGE", xlab="AG
E", ylab="n")
hist(aggregate(data, list(data$Player), max)$Exp, main="Distribution of player's Experience", xl
ab="Experience", ylab="n")
plot(aggregate(data, list(data$Age), mean)$Age, aggregate(data, list(data$Age), mean)$G, type=
'o', main="Avg Games per season depending on AGE", xlab="AGE", ylab="Games")
plot(aggregate(data, list(data$Exp), mean)$Exp, aggregate(data, list(data$Exp), mean)$G, type=
'o', main="Avg Games per season depending on Experience", xlab="Experience", ylab="Games")
```

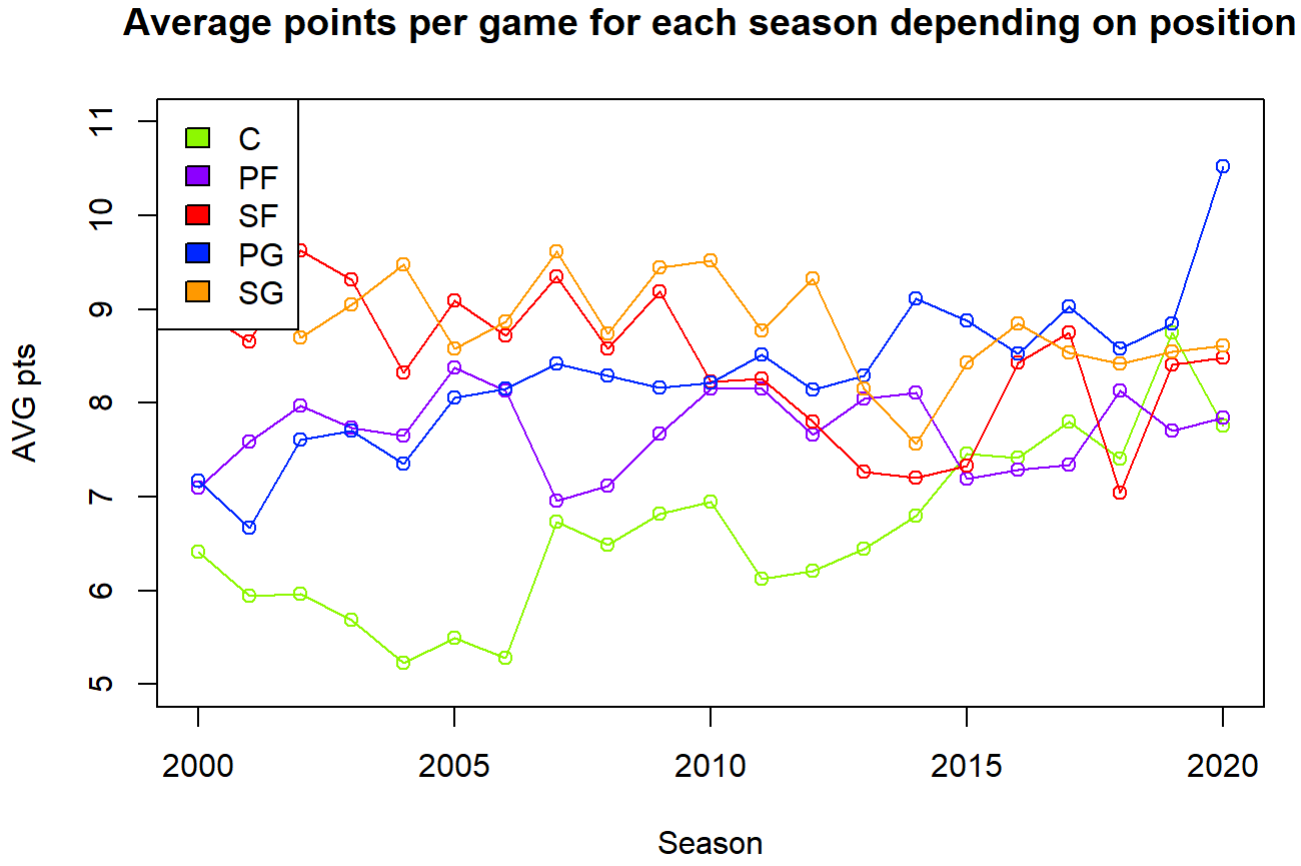








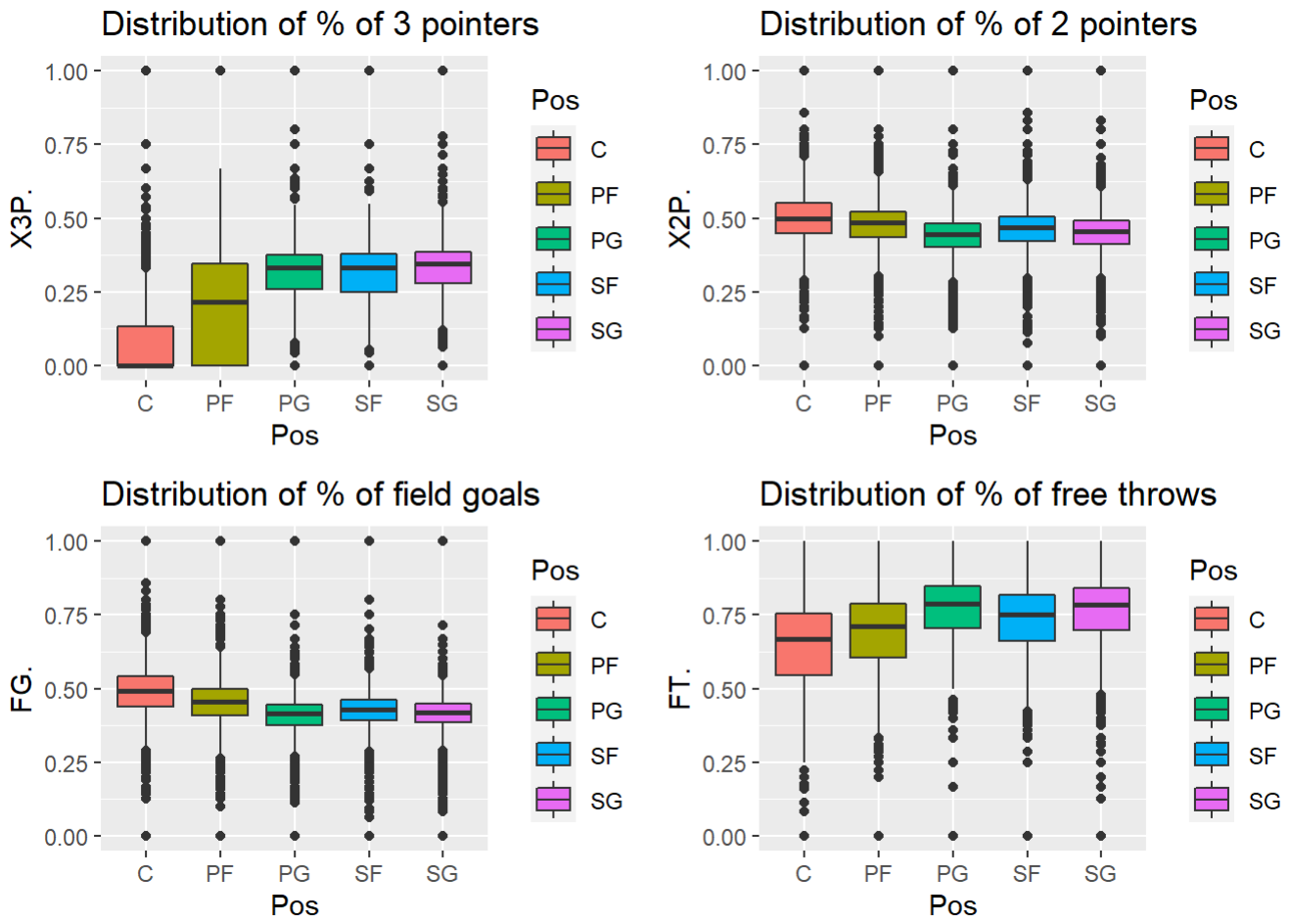

```
c = data[data$Pos == 'C', ]
c = aggregate(c, list(c$Season), mean)$PTS.G
pf = data[data$Pos == 'PF', ]
pf = aggregate(pf, list(pf$Season), mean)$PTS.G
sf = data[data$Pos == 'SF', ]
sf = aggregate(sf, list(sf$Season), mean)$PTS.G
sg = data[data$Pos == 'SG', ]
sg = aggregate(sg, list(sg$Season), mean)$PTS.G
pg = data[data$Pos == 'PG', ]
pg = aggregate(pg, list(pg$Season), mean)$PTS.G

plot(unique(data$Season), c, type='o', col='#8df801', ylim=c(5,11), ylab='AVG pts', xlab='Seaso
n', main="Average points per game for each season depending on position")
lines(unique(data$Season), pf, type='o', col='#8c00ff')
lines(unique(data$Season), sf, type='o', col='#ff0000')
lines(unique(data$Season), pg, type='o', col='#0026ff')
lines(unique(data$Season), sg, type='o', col='#ff9900')
legend(x='topleft',legend=c('C','PF','SF','PG','SG'),fill=c('#8df801','#8c00ff','#ff0000','#0026
ff','#ff9900'))
```



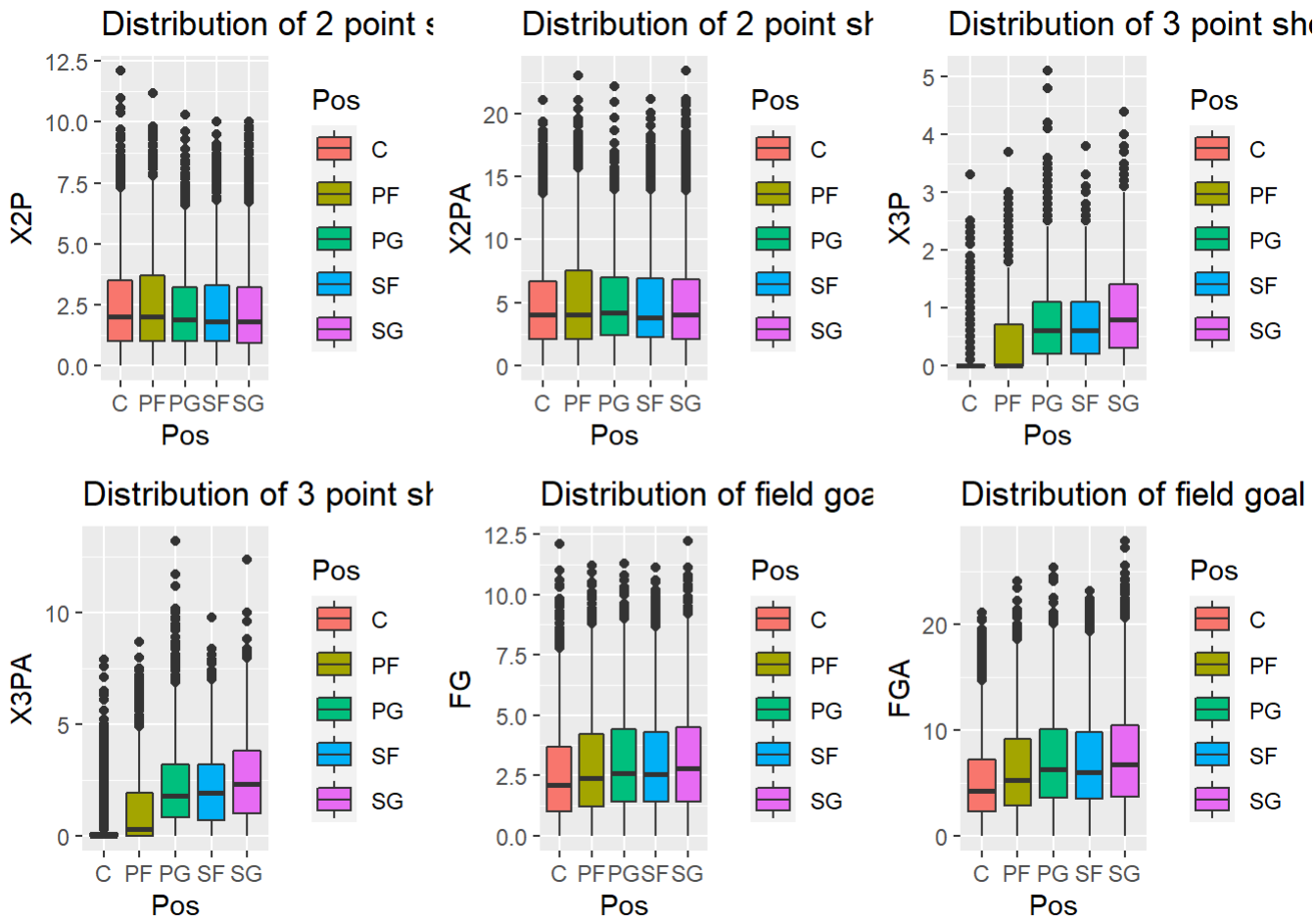
Handling Outliers

Every dataset ought to have outliers, or observations that can decrease the accuracy. Previous analysis shows, that this dataset is pretty “clean”, but still it contains many observations that are beyond 1st or 3rd quartile + 1.5 IQR, that we can see on the boxplots below. I consider this cases as players, who are uprising stars or just had successful season. Besides, each feature has such outliers, so I decided to keep them.

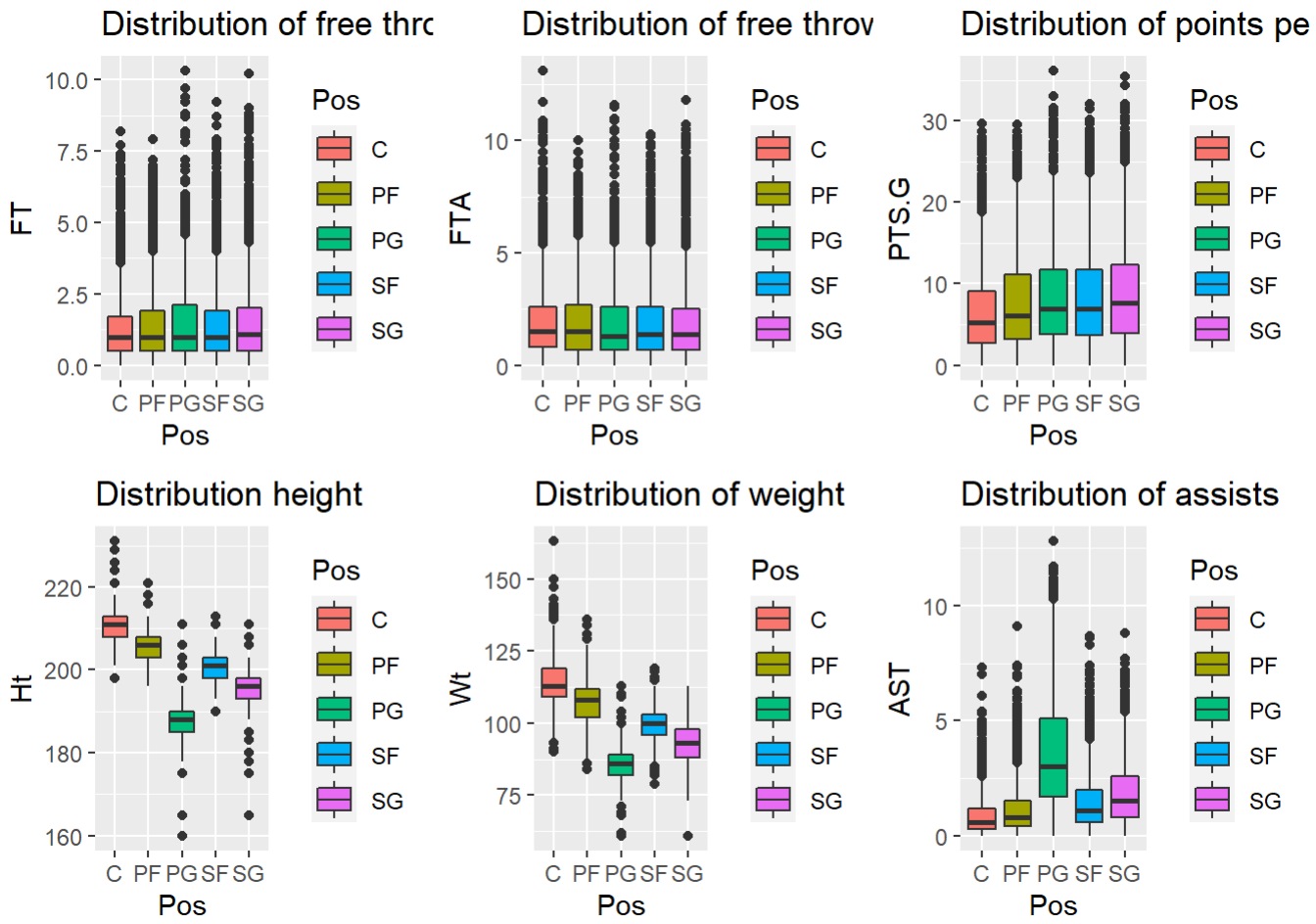
```
x3p <- ggplot(data=data, aes(x=Pos, y=X3P.)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distributi
on of % of 3 pointers")
x2p <- ggplot(data=data, aes(x=Pos, y=X2P.)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distributi
on of % of 2 pointers")
fg <- ggplot(data=data, aes(x=Pos, y=FG.)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution
of % of field goals")
ft <- ggplot(data=data, aes(x=Pos, y=FT.)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution
of % of free throws")
ggarrange(x3p, x2p, fg, ft, ncol=2, nrow=2)
```



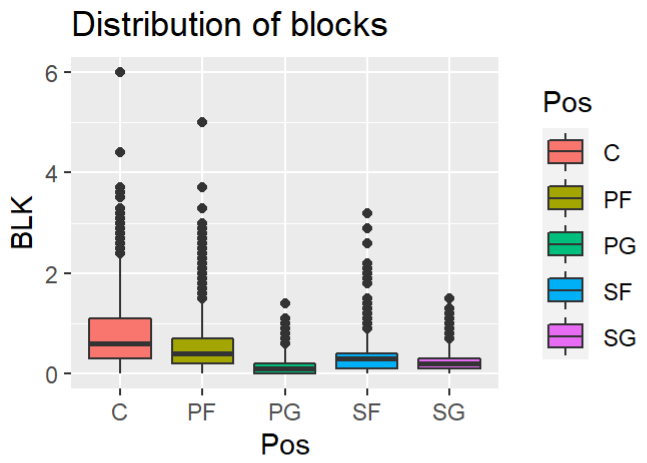

```
ggarrange(ggplot(data=data, aes(x=Pos, y=X2P)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribu
tion of 2 point shots"),
    ggplot(data=data, aes(x=Pos, y=X2PA)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution
 of 2 point shot attempts"),
    ggplot(data=data, aes(x=Pos, y=X3P)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f 3 point shots"),
    ggplot(data=data, aes(x=Pos, y=X3PA)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution
 of 3 point shot attempts"),
    ggplot(data=data, aes(x=Pos, y=FG)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution of
field goals"),
    ggplot(data=data, aes(x=Pos, y=FGA)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f field goal attempts"),
    ncol=3, nrow=2)
```

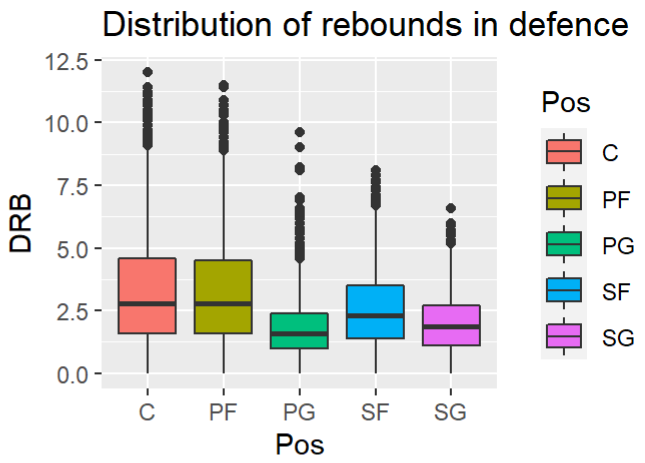


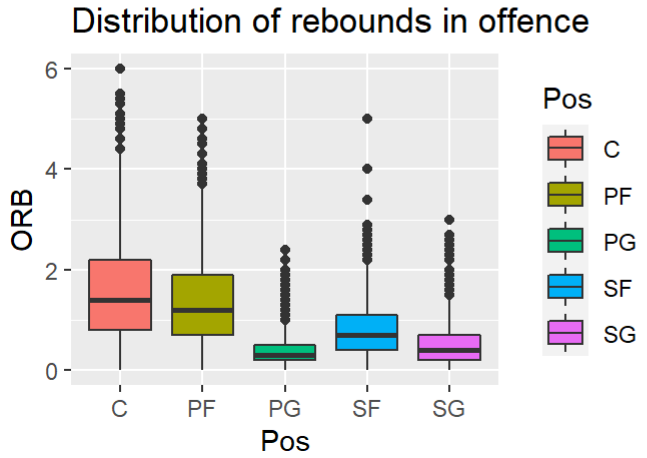

```
ggarrange(ggplot(data=data, aes(x=Pos, y=FT)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribut
ion of free throws"),
    ggplot(data=data, aes(x=Pos, y=FTA)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f free throw attempts"),
    ggplot(data=data, aes(x=Pos, y=PTS.G)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution
of points per game"),
    ggplot(data=data, aes(x=Pos, y=Ht)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution he
ight"),
    ggplot(data=data, aes(x=Pos, y=Wt)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution of
weight"),
    ggplot(data=data, aes(x=Pos, y=AST)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f assists"),
    ncol=3, nrow=2)
```

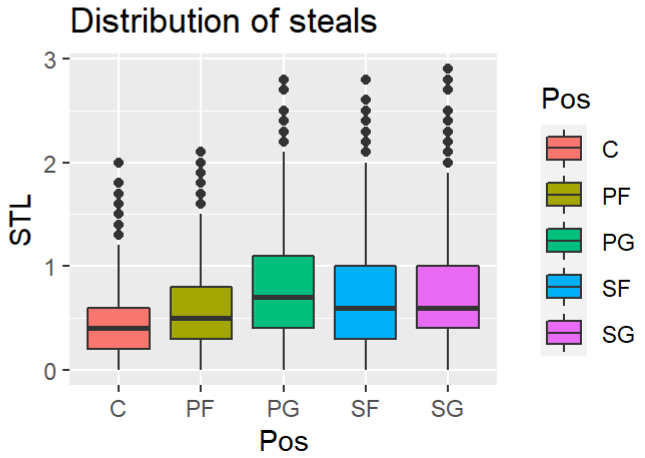



```
ggarrange(ggplot(data=data, aes(x=Pos, y=BLK)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribu
tion of blocks"),
    ggplot(data=data, aes(x=Pos, y=DRB)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f rebounds in defence"),
    ggplot(data=data, aes(x=Pos, y=ORB)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f rebounds in offence"),
    ggplot(data=data, aes(x=Pos, y=STL)) + geom_boxplot(aes(fill=Pos)) + ggtitle("Distribution o
f steals"),
    ncol=2, nrow=2)
```









Feature Encoding

Remember that Machine Learning algorithms perform Linear Algebra on Matrices, which means all features need have numeric values. The process of converting Categorical Features into values is called Encoding. Let's encode all positions to the corresponding numbers.

```
data$Pos <- replace(data$Pos, data$Pos == 'C', 5)
data$Pos <- replace(data$Pos, data$Pos == 'PF', 4)
data$Pos <- replace(data$Pos, data$Pos == 'SF', 3)
data$Pos <- replace(data$Pos, data$Pos == 'SG', 2)
data$Pos <- replace(data$Pos, data$Pos == 'PG', 1)
head(data)
```

	Player <chr>	Pos <chr>	Ht <int>	Wt <int>	Exp <int>	Age <int>	G <int>	GS <int>	MP <dbl>
1	Nick Anderson	2	198	93	10	32	72	72	29.1
2	Jon Barry	2	193	88	7	30	62	1	20.7
3	Tyrone Corbin	3	198	95	14	37	54	5	17.4
4	Tony Delk	1	185	86	3	26	46	1	14.8
5	Vlade Divac	5	216	110	10	31	82	81	29.0

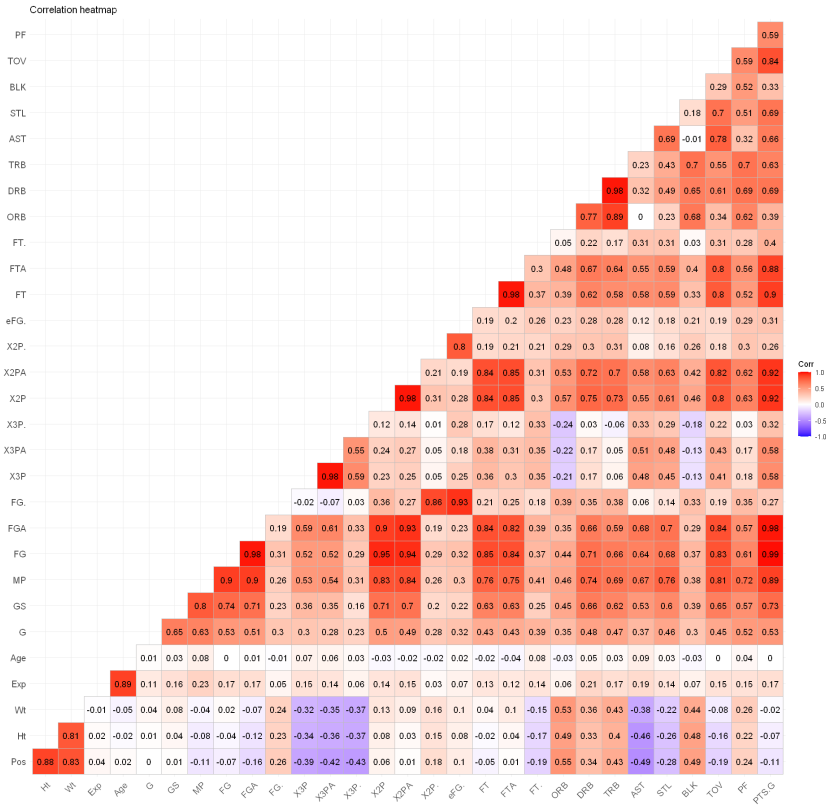
	Player <chr>	Pos <chr>	Ht <int>	Wt <int>	Exp <int>	Age <int>	G <int>	GS <int>	MP <dbl>
6	Lawrence Funderburke	4	206	104	2	29	75	1	13.7

6 rows | 1-10 of 33 columns

Feature Correlation

Correlation ia s measure of how much two random variables change together. Features should be uncorrelated with each other and highly correlated to the feature we're trying to predict.

```
cor_data = select(data, 2:31)

correlations <- cor(cor_data %>% type.convert(as.is=TRUE))
ggcorrplot(correlations, type = "lower", lab=TRUE, )
```



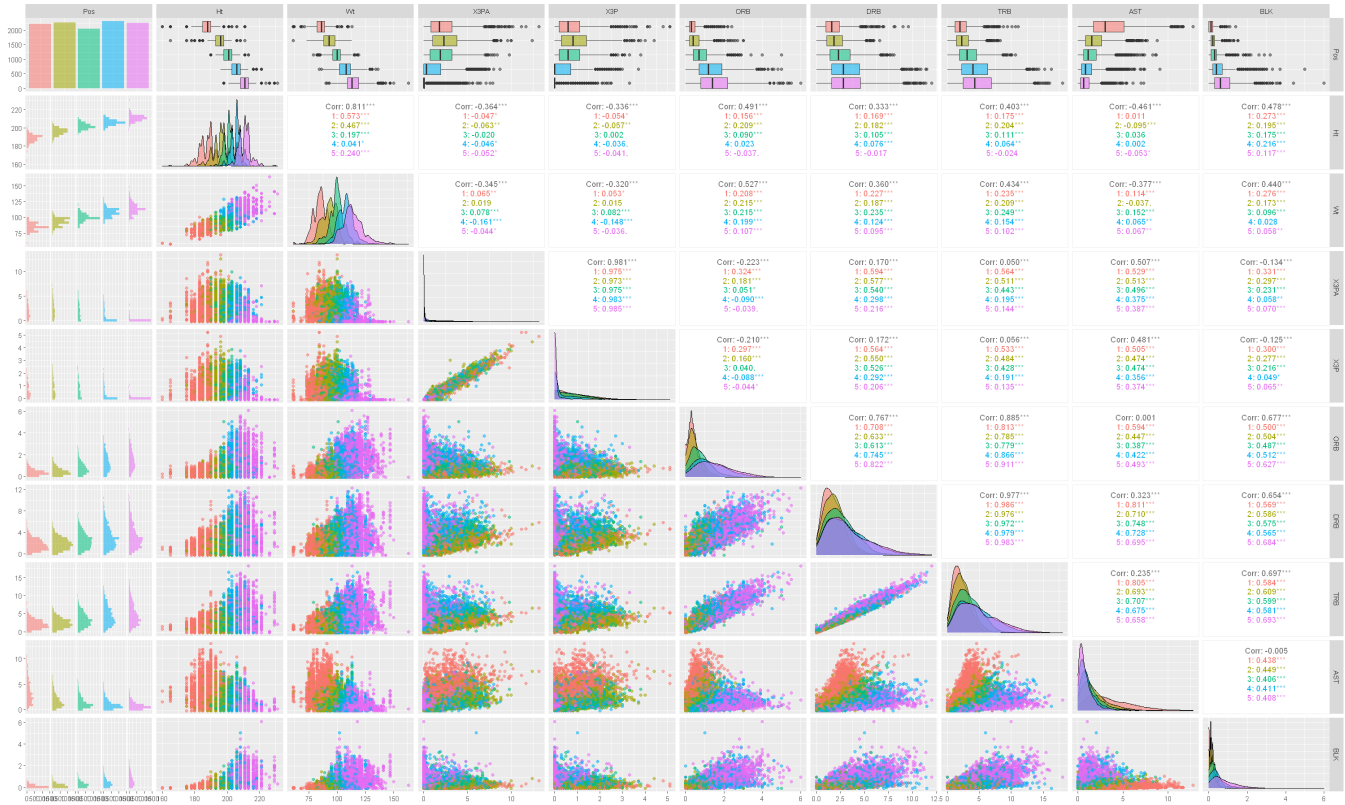
> Filter-based feature selection methods use statistical measures to score the correlation or dependence between input variables that can be filtered to choose the most relevant features.

Source (https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/)

First of all I decided to visually choose features, that correlates with target variable in condition $|corr| > 0.4$. There are 8 of them: Ht, Wt, X3PA, X3P, ORB, TRB, AST, BLK

Let's look at their distributions uni- and bi-variative. The differences among classes should be noticable.

```
attach(data)
ggpairs(select(data, Pos,Ht,Wt,X3PA,X3P,ORB,DRB,TRB,AST,BLK), aes(color=Pos, alpha=0.5))
```



Nice! Then I decided to use filter-based method called ANOVA in order to check their signifficance for the predictiing Pos.

```
attach(data)
```

```
anova <- aov(Pos~Ht*Wt*X3PA*X3P*ORB*DRB*TRB*AST*BLK)
smr <- summary(anova)


coef.tidy <- tidy(anova)
colnames(coef.tidy) <- c("Feature", "Df", "Sum_Sq", "Mean_Sq", "F_value", "pvalue")
coef.tidy[order(coef.tidy$F_value, decreasing = TRUE), ]
```

Feature <chr>	**Df** <dbl>	**Sum_Sq** <dbl>	**Mean_Sq** <dbl>	**F_value** <dbl>	**pv** <
Ht	1	1.744096e+04	1.744096e+04	7.754500e+04	0.000000
Wt	1	7.893640e+02	7.893640e+02	3.509624e+03	0.000000
AST	1	2.065518e+02	2.065518e+02	9.183586e+02	1.655751e
X3PA	1	1.671119e+02	1.671119e+02	7.430030e+02	3.656772e
ORB	1	1.594045e+02	1.594045e+02	7.087350e+02	3.429533e

Feature <chr>	Df <dbl>	Sum_Sq <dbl>	Mean_Sq <dbl>	F_value <dbl>	pv
Ht:ORB	1	1.105264e+02	1.105264e+02	4.914161e+02	1.837527e
Ht:X3PA	1	9.374288e+01	9.374288e+01	4.167941e+02	6.82735
Ht:Wt:AST	1	8.947397e+01	8.947397e+01	3.978139e+02	6.49174
Ht:Wt:X3PA	1	8.614445e+01	8.614445e+01	3.830104e+02	8.32126
Ht:Wt	1	7.468455e+01	7.468455e+01	3.320581e+02	4.48546

1-10 of 512 rows Previous **1** 2 3 4 5 6 … 52 Next

The table above shows all possible compinations of features and their impact on the target value. We assume the hypothesis that none of the features has impact on the target. The smaller p-value is, the bigger chances to reject this hypothesis and accept another one - some of the features have impact. The bigger F score is - the biger is impact.

Unfortunately, the first 5 (top) rows contain only one feature, it means ANOVA recommends to fit the model having only one feature…

Further I will try to fit first 9 combinations.