NBA Player Performance Analysis: PCA, Hierarchical Clustering, and K-Means Clustering

Hassan OUKHOUYA*

2023-04-02

About me

- ***Hassan OUKHOUYA***
- ***E-mail***: hassan.oukhouya@um5r.ac.ma
- ***LinkedIn***: https://www.linkedin.com/in/hassan-oukhouya-3901b816b/
- ***ORCID iD***: https://orcid.org/0000-0002-5058-2008
- ***Upwork***: https://www.upwork.com/services/product/time-series-analysis-with-python-or-r-studio-1449669530698514432?ref=project_share
- **Fiverr**: https://www.fiverr.com/share/5Dzz6z

Introduction

Data science and sports analytics have opened new doors for using machine learning and data mining techniques in basketball performance analysis. This study aims to explore the advanced measures of basketball performance in the National Basketball Association (NBA) using Principal Component Analysis (PCA) and Clustering. PCA identifies the most significant variables, while Clustering groups similar observations using unsupervised classification algorithms. In this project, we will analyze NBA player data using PCA and Clustering methods in R to evaluate player performance and help coaches make better decisions. Our results will be presented using graphs and tables to visualize the clusters' impact on the data.

Problematic

In this case study, we will examine NBA data to explore and understand several key questions. Specifically, we will seek to determine:

1) The best NBA players of all time based on a variety of basketball skills and not just points?
2) If NBA players possess skills that exceed their designated position, how can we help NBA coaches make better decisions and evaluate whether they are making the right choices?

Objective of the study

Data description

The data set contains the performances of the most important players in the history of NBA games. It was collected from the NBA statistics website. You can see the site and the glossary here: The best players in the NBA

This data was created by NBA.com to comprehensively describe the types of shots players take and how they score. The dataset includes all players who played one minute of game time (we'll specify that in one minute). The exact columns are shown below with a sample of a few rows of data.

*Ph.D. student in Time Series Forecasting, LMSA, Department of Mathematics, Faculty of Sciences Rabat Mohammed V University, Morocco, hassan\protect_oukhouya@um5.ac.ma

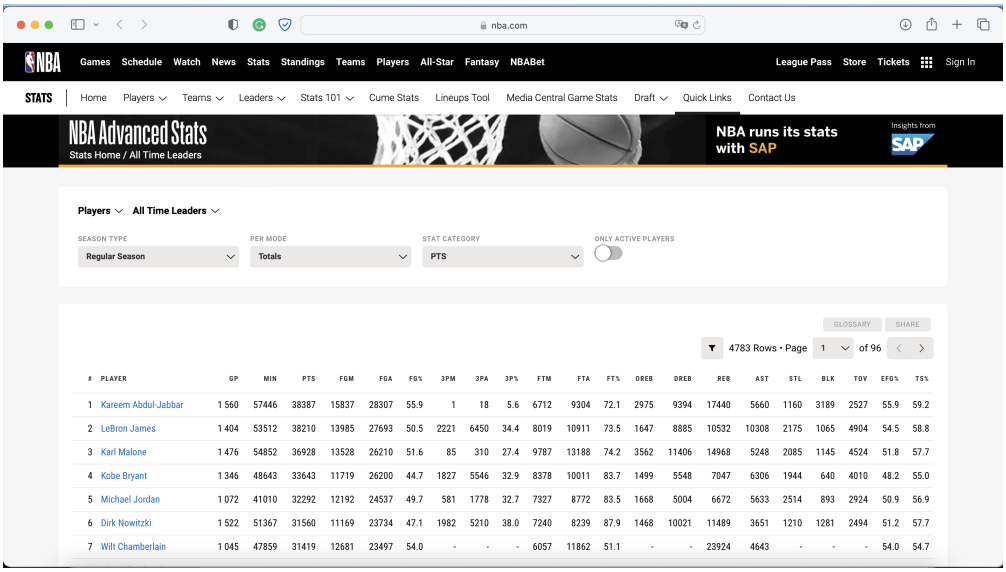


Figure 1: Statistiques avancées de la NBA / Leaders de tous les temps.

Importation of booksellers required

```
#Remove all objects from the workspace
rm(list=ls())

warning=FALSE
message=FALSE

# Import the necessary libraries for cleaning and analysis:

library(tidyverse)
library(GGally) # Visualization based on ggplot2
library(factoextra) # Visualization of the PCA based on ggplot2
library("data.table")
library(dendextend)
library(ggfortify)
library(ggrepel)
library(gridExtra)
library(knitr)
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 3 > 1' in coercion to 'logical(1)'
```

```
library(mclust)
library(NbClust)
library(plyr)
library(tidyverse)
```

Import and prepare data

To facilitate the task of collecting the data, here is the link to the data we will use in this work on the Kaggle site: Top NBA Players. Installation of the `kableExtra` library for a good visualization of the data set table.

```r
# Installation of the library
#install.packages("kableExtra")
# Loading libraries
library(knitr)
library(kableExtra)
```

```r
# Import data
historical_players.df = read.csv(file = "nba.csv", header=T, sep=";")
```

We now see the structure of this data set:

```r
glimpse(historical_players.df)
```

```
## Rows: 1,214
## Columns: 26
## $ player_id   <int> 893, 76375, 76127, 201142, 2544, 78497, 947, 77847, 76804,~
## $ player_name <chr> "Michael Jordan", "Wilt Chamberlain", "Elgin Baylor", "Kev~
## $ gp          <int> 1072, 1045, 846, 703, 1061, 932, 914, 792, 791, 1040, 1476~
## $ min         <dbl> 38.25653, 45.79809, 40.02719, 37.37980, 38.90009, 39.23927~
## $ fgm         <dbl> 11.37313, 12.13493, 10.27541, 9.19346, 9.82375, 9.67382, 9~
## $ fga         <dbl> 22.88899, 22.48517, 23.84279, 18.85349, 19.60697, 20.42060~
## $ fg_pct      <dbl> 0.497, 0.540, 0.431, 0.488, 0.501, 0.474, 0.425, 0.436, 0.~
## $ fg3m        <dbl> 0.54198, NA, NA, 1.79232, 1.38266, NA, 1.15864, NA, 0.0973~
## $ fg3a        <dbl> 1.65858, NA, NA, 4.72404, 4.04807, NA, 3.70131, NA, 0.3274~
## $ fg3_pct     <dbl> 0.327, NA, NA, 0.379, 0.342, NA, 0.313, NA, 0.297, NA, 0.2~
## $ ftm         <dbl> 6.83489, 5.79617, 6.81206, 7.01991, 6.10179, 7.68240, 6.97~
## $ fta         <dbl> 8.18284, 11.35120, 8.73641, 7.96017, 8.24882, 9.44313, 8.9~
## $ ft_pct      <dbl> 0.835, 0.511, 0.780, 0.882, 0.740, 0.814, 0.780, 0.761, 0.~
## $ oreb        <dbl> 1.55597, NA, NA, 0.78663, 1.21489, 0.96774, 0.81510, NA, 1~
## $ dreb        <dbl> 4.66791, NA, NA, 6.36984, 6.04807, 2.77419, 2.89825, NA, 3~
## $ reb         <dbl> 6.22388, 22.89378, 13.54965, 7.15647, 7.26296, 5.76824, 3.~
## $ ast         <dbl> 5.25466, 4.44306, 4.31442, 3.78805, 7.03205, 6.69313, 6.15~
## $ stl         <dbl> 2.34515, NA, NA, 1.19488, 1.64844, 2.61290, 2.16958, NA, 1~
## $ blk         <dbl> 0.83302, NA, NA, 1.04979, 0.77003, 0.74194, 0.17943, NA, 0~
## $ tov         <dbl> 2.72761, NA, NA, 3.15932, 3.41093, NA, 3.56893, NA, 3.0141~
## $ pf          <dbl> 2.59608, 1.98565, 3.06856, 1.89189, 1.86334, 2.61266, 1.94~
## $ pts         <dbl> 30.12313, 30.06603, 27.36288, 27.19915, 27.13195, 27.03004~
## $ ast_tov     <dbl> 1.92647, NA, NA, 1.19901, 2.06162, NA, 1.72410, NA, 0.9246~
## $ stl_tov     <dbl> 0.85978, NA, NA, 0.37821, 0.48328, NA, 0.60791, NA, 0.3912~
## $ efg_pct     <dbl> 0.50872, 0.53969, 0.43097, 0.53516, 0.53629, 0.47373, 0.45~
## $ ts_pct      <dbl> 0.56859, 0.54706, 0.49415, 0.60832, 0.58382, 0.54994, 0.51~
```

We can see that the first two variables `player_id` and `gp` are integers and the variable `player_name` is a character and the other variables are real.

```r
head(historical_players.df) %>%
  kable("latex", booktabs = T,
        caption = "The first observations of our dataset.
        \\label{tab:tableau1}")%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

Table 1: The first observations of our dataset.

player_id	player_name	gp	min	fgm	fga	fg_pct	fg3m	fg3a	fg3_pct	ftm	fta	ft_pct	oreb	dreb	reb	ast	stl	blk	tov	pf	pts	ast_tov	stl_tov	efg_pct	ts_pct
893	Michael Jordan	1072	38.25653	11.37313	22.88899	0.497	0.54198	1.65858	0.327	6.83489	8.18284	0.835	1.55597	4.66791	6.22388	5.25466	2.34515	0.83302	2.72761	2.59608	30.12313	1.92647	0.85978	0.50872	0.56859
76375	Wilt Chamberlain	1045	45.79809	12.13493	22.48517	0.540	NA	NA	NA	5.79617	11.35120	0.511	NA	NA	22.80378	4.44306	NA	NA	NA	1.98565	30.06603	NA	NA	0.53969	0.54706
76127	Elgin Baylor	846	40.02719	10.27541	23.84279	0.431	NA	NA	NA	6.81206	8.73641	0.780	NA	NA	13.54965	4.31442	NA	NA	NA	3.06856	27.36288	NA	NA	0.43097	0.49415
201142	Kevin Durant	703	37.37980	9.19346	18.85349	0.488	1.79232	4.72404	0.379	7.01991	7.96017	0.882	0.78663	6.36984	7.15647	3.78805	1.19488	1.04979	3.15932	1.89189	27.19915	1.19901	0.37821	0.53516	0.60832
2544	LeBron James	1061	38.90009	9.82375	19.60697	0.501	1.38266	4.04807	0.342	6.10179	8.24882	0.740	1.21489	6.04807	7.26296	7.03205	1.64844	0.77003	3.41093	1.86334	27.13195	2.06162	0.48328	0.53629	0.58382
78497	Jerry West	932	39.23927	9.67382	20.42060	0.474	NA	NA	NA	7.68240	9.44313	0.814	0.96774	2.77419	5.76824	6.69313	2.61290	0.74194	NA	2.61266	27.03004	NA	NA	0.47373	0.54994

```
tail(historical_players.df) %>%
  kable("latex", booktabs = T,
        caption = "The last observations of our dataset.")%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

Table 2: The last observations of our dataset.

	player_id	player_name	gp	min	fgm	fga	fg_pct	fg3m	fg3a	fg3_pct	ftm	fta	ft_pct	oreb	dreb	reb	ast	stl	blk	tov	pf	pts	ast_tov	stl_tov	efg_pct	ts_pct
1209	279	Charles Jones	726	19.41460	0.99862	2.07989	0.480	0.00000	0.00964	0.000	0.51791	0.83747	0.618	1.53168	2.94904	4.48072	0.85813	0.60882	1.56198	0.67080	2.86364	2.51515	1.27926	0.90760	0.48013	0.51364
1210	201202	Joel Anthony	490	14.40000	0.81224	1.58367	0.513	0.00000	0.00000	0.000	0.54694	0.82653	0.662	1.14694	1.61429	2.76122	0.20204	0.25918	1.08367	0.46327	1.67143	2.17143	0.43612	0.55947	0.51289	0.55754
1211	2058	Mark Madsen	453	11.77704	0.84106	1.84106	0.457	0.00221	0.03532	0.063	0.46799	0.88742	0.527	1.23400	1.32009	2.55408	0.39956	0.25607	0.22517	0.45475	1.91170	2.15232	0.87864	0.56311	0.45743	0.48225
1212	1068	Greg Dreiling	474	8.94726	0.79114	1.69409	0.467	0.00633	0.01899	0.333	0.55063	0.84810	0.649	0.60549	1.54219	2.14768	0.39241	0.17300	0.28903	0.52743	1.94726	2.13924	0.74400	0.32800	0.46887	0.51741
1213	2205	DeSagana Diop	601	13.98170	0.83694	1.95840	0.427	0.00166	0.00998	0.167	0.29617	0.63394	0.467	1.38769	2.30449	3.69218	0.44925	0.37770	1.04825	0.59068	2.02829	1.97171	0.76056	0.63944	0.42778	0.44064
1214	1913	Michael Ruffin	414	14.41063	0.62319	1.53140	0.407	0.00000	0.00725	0.000	0.48309	1.05314	0.459	1.76087	2.18116	3.94203	0.60386	0.48068	0.46618	0.65942	2.27536	1.72947	0.91575	0.72894	0.40694	0.43350

Let us now briefly describe the data set. The data set consists of 1214 observations with 26 variables. However, this amount of data is too large for the purpose of this analysis. We will also perform a variable selection by choosing 26 of the available variables. The selected variables are:

Code - Designation	Code - Designation	Code - Designation
`player_id`: Player's id	`fg3a`: 3 Point Field Goals Attempted	`efg_pct`: Effective Field Goal Percentage
`player_name`: Player's name	`oreb`: Offensive Rebounds	`ts_pct`: True Shooting Percentage
`gp`: Games Played	`dreb`: Defensive Rebounds	`fta`: Free Throws Attempted
`min`: Minutes Played	`reb`: Rebounds	`ft-pct`: Free Throw Percentage
`fgm`: Field Goals Made	`ast`: Assists	`fg3_pct`: 3 Point Field Goal Percentage
`fga`: Field Goals Attempted	`stl`: Steals	`pts`: Points
`fg_pct`: Field Goal Percentage	`blk`: Blocks	`ast_tov`: Assist to Turnover Ratio
`fg3m`: 3 Point Field Goals Made	`tov`: Turnovers	`stl_tov`: Steals to turnover
`pf`: Personal Fouls		

Variables	Description
`player_id`	The identification number of each player.
`player_name`	The name of each player.
`gp`	The number of games played.
`min`	The number of minutes played by a player or a team.
`fgm`	The percentage of a team's goals that a player has made on the field.
`fga`	The percentage of the team's field goals a player attempted while on the field.
`fg_pct`	The percentage of shot attempts a player makes on goal. Its formula is: $\frac{\text{fgm}}{\text{fga}}$
`fg3m`	The number of 3-point goals a player or team has made.
`fg3a`	The number of 3-point field goals a player or team has attempted.
`fg3_pct`	The percentage of 3-point field goal attempts a player makes. Its formula is: $\frac{\text{fg3m}}{\text{fg3a}}$
`fta`	The number of free throws a player or team has attempted.

Variables	Description
ft-pct	The percentage of free throw attempts that a player or team has made.
oreb	The number of rebounds a player or team collected while on offense.
dreb	The number of rebounds a player or team collected while on defense.
reb	Bounces offered by a player or team while they were on the court.
ast	The number of assists (passes that lead directly to a scored basket) made by a player.
stl	The number of times a defensive player or team takes the ball from an offensive player causes a turnover.
blk	A block occurs when an offensive player attempts a shot, and the defensive player knocks the ball away, blocking their chance to score.
tov	A turnover occurs when the attacking player or team loses the ball to the defense.
pf	The number of personal fouls committed by a player or a team.
pts	The number of points scored.
ast_tov	The number of assists for a player or team compared to the number of turnovers they committed.
stl_tov	Team losses are credited to the defensive player who caused the turnover.
efg_pct	Measures field goal percentage, taking into account that 3-point field goals are 1.5 times more valuable than 2-point field goals. Its formula is: $\frac{(\text{fgm}+(0.5\times\text{fg3m})}{\text{fga}}$
ts_pct	A percentage shot that takes into account the value of three-point field goals and free throws in addition to conventional two-point field goals. Its formula is: $\frac{\text{pts}}{2\times(\text{fga}+0.44\times\text{fta})}$

Each variable was converted to a "per game" measure (e.g., points per game calculated as the number of points scored divided by the total games played) for normalization. From the data structure output and the table's initial observations (1), it is clear that there are missing values, so we need to deal with them later.

Data cleaning

Missing values

We performed a missing value analysis and treatment of the data set:

```
# Indicate observed and missing values
#is.na(historical_players.df)
```

Identify the position of columns with at least one missing value:

```
# check for missing values on each column (variable)
which(colSums(is.na(historical_players.df))>0)
```

```
##    fg3m    fg3a fg3_pct    oreb    dreb     stl     blk     tov ast_tov stl_tov
##       8       9      10      14      15      18      19      20      23      24
```

Count the number of missing values per column:

```
colSums(is.na(historical_players.df))
```

```
##   player_id player_name          gp         min         fgm         fga
##           0           0           0           0           0           0
##      fg_pct        fg3m        fg3a    fg3_pct         ftm         fta
##           0         212         212         212           0           0
##      ft_pct        oreb        dreb         reb         ast         stl
##           0         119         119           0           0         119
##         blk         tov          pf         pts     ast_tov     stl_tov
```

```
##          119          187            0            0          187          187
##     efg_pct      ts_pct
##           0            0
```

To view missing values in a table:

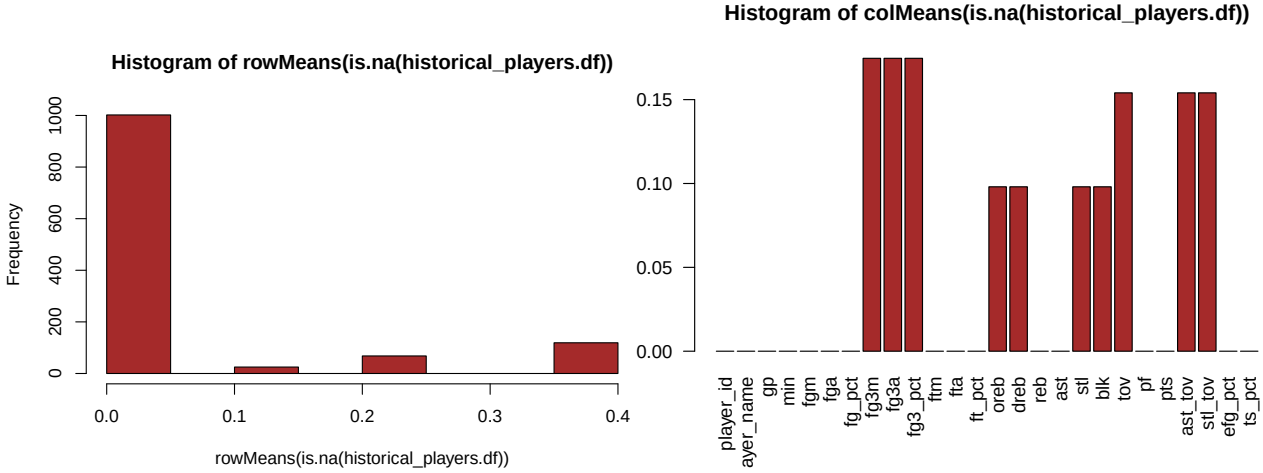
```
kable(historical_players.df %>%
  select(everything()) %>%
  summarise_all(funs(sum(is.na(.)))))%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

player_id	player_name	gp	min	fgm	fga	fg_pct	fg3m	fg3a	fg3_pct	ftm	fta	ft_pct	oreb	dreb	reb	ast	stl	blk	tov	pf	pts	ast_tov	stl_tov	efg_pct	ts_pct
0	0	0	0	0	0	0	212	212	212	0	0	0	119	119	0	0	119	119	187	0	0	187	187	0	0

We visualize the missing values graphically on histograms. We will visualize the average of the missing values according to the observations (rows) and the variables (columns), to see the distribution of the missing values in the bar plots:

```
hist(rowMeans(is.na(historical_players.df)),col = "Brown")

barplot(colMeans(is.na(historical_players.df)), las=2,col = "Brown",
        main = "Histogram of colMeans(is.na(historical_players.df))")
```



According to the missing values table above and the two histograms, we see that there are some columns, namely `fg3m`, `fg3a`, `fg3_pct`, `oreb`, `dreb`, `stl`, `blk`, `tov`, `ast_tov`, `stl_tov` with several missing values respectively 212, 212, 212, 119, 119, 119, 187, 187 There are several methods to deal with missing values, among them deletion, replacement by the mean or linear interpolation... etc. But to choose one of these methods, it is necessary to understand the reason for the existence of the missing values through an analysis. In our case, we will only remove the "NA" lines. These lines are redundant.

```
# Skip players whose information is NA in any variable
historical_players.df =
  historical_players.df[rowSums(is.na(historical_players.df)) == 0,]
dim(historical_players.df)
```

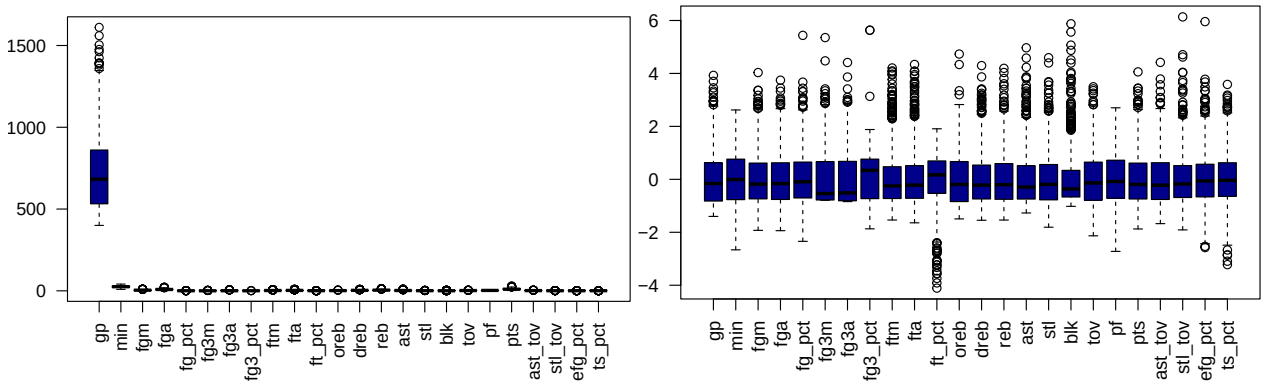
```
## [1] 1002   26
```

We now have 1002 observations and 26 variables cleaned by missing values.

Outliers

Box and bar graphs (box plots) or boxplots represent statistical data visually. They allow you to visualize variations in the data distribution and general trends. We can now check for outliers on each variable in the previous data cleaning by using boxplots:

```
# We eliminate the first two variables because one is id and the other qualitative
boxplot(historical_players.df[, -2:-1], las=2, col="darkblue")
# To scale or not?
boxplot(scale(historical_players.df[, -2:-1]), las=2, col="darkblue")
```



We notice outliers on the box plots of the 24 variables in our dataset. These values can be considered outliers and can have a significant impact on the results of the analysis. Understanding the cause of these outliers is critical to avoid drawing incorrect conclusions. These outliers should be checked to ensure that they are not errors - they may be real cases - and to understand their impact on the analysis results.

We will identify outliers for each variable:

```
boxplot.stats(historical_players.df$gp)$out
```

```
##  [1] 1476 1560 1394 1392 1389 1462 1611 1380 1359 1504 1391 1424
```

```
boxplot.stats(historical_players.df$min)$out
```

```
## numeric(0)
```

```
boxplot.stats(historical_players.df$fgm)$out
```

```
##  [1] 11.37313  9.19346  9.82375  9.26368 10.17067  9.16531  9.27654 10.15192
##  [9]  9.57748  9.40274  9.38691  9.13350  8.95881  8.93462
```

```
boxplot.stats(historical_players.df$fga)$out
```

```
##  [1] 22.88899 19.60697 21.77899 19.90771 19.46508 20.10149 19.54098 19.32441
##  [9] 21.31459 20.35642
```

```
boxplot.stats(historical_players.df$fg_pct)$out
```

```
##  [1] 0.582 0.585 0.599 0.572 0.572 0.677 0.594 0.571 0.574 0.608 0.594 0.580
## [13] 0.584
```

```
boxplot.stats(historical_players.df$fg3m)$out
```

```
##  [1] 3.33972 2.24715 2.86207 2.28692 2.00223 2.18905 2.07927 2.11486 2.12950
## [10] 2.01591 1.98739
```

```
boxplot.stats(historical_players.df$fg3a)$out
```

```
## [1] 7.62892 6.18049 5.57428 6.83405 5.71462 5.46269 5.46748 5.51520 5.58813
```

```
boxplot.stats(historical_players.df$fg3_pct)$out
```

```
## [1] 0.667 1.000 1.000
```

```
boxplot.stats(historical_players.df$ftm)$out
```

```
##  [1] 6.83489 7.01991 6.10179 6.97484 5.74083 6.63076 6.22437 5.61546 5.89344
## [10] 7.15393 5.41641 4.91715 5.91038 5.94611 6.87154 5.91705 5.53045 5.09613
## [19] 5.54004 6.11449 5.84438 5.21558 6.41911 5.28134 5.32506 5.15115 5.47461
## [28] 5.17311 5.00112 5.11317 5.00182 4.89598 5.32420 4.83507 5.36463 4.89623
## [37] 4.98739 4.84129 5.56832
```

```
boxplot.stats(historical_players.df$fta)$out
```

```
##  [1] 8.18284 7.96017 8.24882 8.93654 6.80531 8.93496 7.43759 6.92551 7.24898
## [10] 8.74450 6.60182 9.32229 7.69508 7.22904 8.05041 8.05499 7.10413 7.01699
## [19] 7.52977 8.30902 7.81326 6.49275 8.34462 6.62396 7.06998 6.38719 6.45695
## [28] 6.69708 6.43499 6.57534 6.38231 8.65304 6.74053 6.34543 6.99523 6.77025
```

```
boxplot.stats(historical_players.df$ft_pct)$out
```

```
##  [1] 0.527 0.430 0.535 0.531 0.500 0.414 0.522 0.529 0.521 0.554 0.538 0.553
## [13] 0.498 0.485 0.478 0.499 0.528 0.458 0.444 0.494 0.551 0.486 0.527 0.467
## [25] 0.459
```

```
boxplot.stats(historical_players.df$oreb)$out
```

```
## [1] 3.97018 5.06471 4.75192 3.85164
```

```
boxplot.stats(historical_players.df$dreb)$out
```

```
##  [1] 7.72764 7.58192 8.04571 7.36537 7.72227 7.12694 7.84653 7.90554 7.51165
## [10] 7.48521 7.79978 7.04332 7.13394 8.06897 7.33577 8.38194 7.83379 9.77481
## [19] 9.11635 7.17492 7.14866 7.47425 8.16239 7.09831 7.68712 8.41624 7.14967
## [28] 7.09353 7.00000 8.36992
```

```
boxplot.stats(historical_players.df$reb)$out
```

```
##  [1] 11.17949 10.85253 11.69245 11.10501 10.81520 10.63526 12.49348 12.19865
##  [9] 10.84124 11.47917 13.63446 12.67191 10.51923 10.83231 13.99289 13.12184
```

```
boxplot.stats(historical_players.df$ast)$out
```

```
##  [1]  7.03205  6.82404  7.92365 11.19316  7.64894  9.25536  9.22000  7.39269
##  [9]  9.89329  9.13061  8.18339  6.71610  8.06982  7.21557  8.31510  6.73546
## [17]  7.01259  8.49219  6.71766  7.30073 10.50931  8.69231  8.06373  6.71390
## [25]  6.95474  8.50840  7.97377  7.43462  7.56580
```

```
boxplot.stats(historical_players.df$stl)$out
```

```
##  [1] 2.34515 2.16958 1.99278 2.03223 1.90287 1.90092 1.89694 2.29257 1.98545
## [10] 1.95840 2.63129 2.71117 2.21543 2.33408 2.17088 1.92955 2.09809 1.95577
## [19] 1.93973 1.93970
```

. . . etc.

> We see that they all correspond to values that correspond to real players. This means that these are values that players can achieve, so we will not remove these outliers.

Variable selection

After cleaning the data, we will select a subset of all the variables to continue the analysis. Nevertheless, the data cleaning performed in the previous section can be useful for further analysis of the whole data set.

```
nba = historical_players.df[,3:ncol(historical_players.df)]
nba = as.data.frame(sapply(nba, as.numeric ))
names = historical_players.df[,2]

dim(nba)
```

```
## [1] 1002   24
```

```
# The variables we will keep
names(nba)
```

```
##  [1] "gp"      "min"     "fgm"     "fga"     "fg_pct"  "fg3m"    "fg3a"
##  [8] "fg3_pct" "ftm"     "fta"     "ft_pct"  "oreb"    "dreb"    "reb"
## [15] "ast"     "stl"     "blk"     "tov"     "pf"      "pts"     "ast_tov"
## [22] "stl_tov" "efg_pct" "ts_pct"
```

> We will eliminate the first two variables, `palyer_id` and `player_name`, in our clean dataset since the first one contains no training and the other is qualitative; we will select the 24 quantitative variables.

Data analysis, exploration and visualization

In this section, we will conduct an exploratory analysis with the dataset we constructed in the previous sections. To better understand the dataset, we will use.

Descriptive statistics

We can calculate the minimum, first quartile, median, mean, third quartile, and a maximum of all numeric variables in our data set using `summary()` :

```
summary(nba)
```

```
##        gp              min              fgm               fga
##  Min.   : 400.0   Min.   : 8.947   Min.   : 0.6232   Min.   : 1.531
##  1st Qu.: 532.5   1st Qu.:20.489   1st Qu.: 2.7673   1st Qu.: 5.963
##  Median : 682.5   Median :25.101   Median : 3.7753   Median : 8.220
##  Mean   : 718.5   Mean   :25.163   Mean   : 4.0994   Mean   : 8.819
##  3rd Qu.: 861.0   3rd Qu.:29.802   3rd Qu.: 5.2043   3rd Qu.:11.172
##  Max.   :1611.0   Max.   :41.120   Max.   :11.3731   Max.   :22.889
##      fg_pct            fg3m              fg3a            fg3_pct
##  Min.   :0.3730   Min.   :0.000000   Min.   :0.00000   Min.   :0.0000
```

```
##  1st Qu.:0.4370   1st Qu.:0.006258   1st Qu.:0.04473   1st Qu.:0.1520
##  Median :0.4610   Median :0.137780   Median :0.47784   Median :0.2945
##  Mean   :0.4645   Mean   :0.430171   Mean   :1.22112   Mean   :0.2492
##  3rd Qu.:0.4900   3rd Qu.:0.793380   3rd Qu.:2.20637   3rd Qu.:0.3510
##  Max.   :0.6770   Max.   :3.339720   Max.   :7.62892   Max.   :1.0000
##       ftm              fta             ft_pct            oreb
##  Min.   :0.2141   Min.   :0.2441   Min.   :0.4140   Min.   :0.1456
##  1st Qu.:1.1963   1st Qu.:1.6486   1st Qu.:0.7060   1st Qu.:0.6621
##  Median :1.7713   Median :2.4028   Median :0.7630   Median :1.1722
##  Mean   :2.0709   Mean   :2.7368   Mean   :0.7492   Mean   :1.3260
##  3rd Qu.:2.6476   3rd Qu.:3.5213   3rd Qu.:0.8060   3rd Qu.:1.8559
##  Max.   :7.1539   Max.   :9.3223   Max.   :0.9050   Max.   :5.0647
##       dreb             reb              ast               stl
##  Min.   :0.7673   Min.   : 1.007   Min.   : 0.1975   Min.   :0.1225
##  1st Qu.:2.0071   1st Qu.: 2.781   1st Qu.: 1.1235   1st Qu.:0.5405
##  Median :2.8147   Median : 4.031   Median : 1.9202   Median :0.7761
##  Mean   :3.1508   Mean   : 4.492   Mean   : 2.4371   Mean   :0.8537
##  3rd Qu.:3.9814   3rd Qu.: 5.833   3rd Qu.: 3.3430   3rd Qu.:1.0794
##  Max.   :9.7748   Max.   :13.993   Max.   :11.1932   Max.   :2.7112
##       blk               tov              pf              pts
##  Min.   :0.00996   Min.   :0.1734   Min.   :0.7944   Min.   : 1.729
##  1st Qu.:0.18856   1st Qu.:1.0697   1st Qu.:1.9307   1st Qu.: 7.144
##  Median :0.34270   Median :1.5038   Median :2.2904   Median : 9.769
##  Mean   :0.52666   Mean   :1.5942   Mean   :2.3360   Mean   :10.700
##  3rd Qu.:0.69759   3rd Qu.:2.0263   3rd Qu.:2.7452   3rd Qu.:13.634
##  Max.   :3.50171   Max.   :3.9222   Max.   :3.8665   Max.   :30.123
##     ast_tov          stl_tov          efg_pct           ts_pct
##  Min.   :0.2445   Min.   :0.1320   Min.   :0.4069   Min.   :0.4294
##  1st Qu.:0.9114   1st Qu.:0.4061   1st Qu.:0.4675   1st Qu.:0.5105
##  Median :1.3036   Median :0.5208   Median :0.4866   Median :0.5295
##  Mean   :1.4672   Mean   :0.5600   Mean   :0.4885   Mean   :0.5307
##  3rd Qu.:1.9264   3rd Qu.:0.6748   3rd Qu.:0.5066   3rd Qu.:0.5504
##  Max.   :4.6936   Max.   :1.9345   Max.   :0.6771   Max.   :0.6433
```

If we need more descriptive statistics, we use `stat.desc()` from the {pastecs} package:

```
library(pastecs)
```

```
##
## Attaching package: 'pastecs'

## The following objects are masked from 'package:data.table':
##
##     first, last

## The following objects are masked from 'package:dplyr':
##
##     first, last

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
stat.desc(nba) %>%
  kable("latex", booktabs = T,
        caption = "Descriptive statistics.")%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

Table 5: Descriptive statistics.

	gp	min	fgm	fga	fg_pct	fg3m	fg3a	fg3_pct	ftm	fta	ft_pct	oreb	dreb	reb	ast	stl	blk	tov	pf	pts	ast_tov	stl_tov	efg_pct	ts_pct
nbr.val	1.002000e+03	1.002000e+03	1002.0000000	1002.0000000	1.002000e+03	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1002.0000000	1.002000e+03	1002.0000000	1002.0000000	1.002000e+03	1002.0000000
nbr.null	0.000000e+00	0.000000e+00	0.0000000	0.0000000	0.000000e+00	111.0000000	14.0000000	111.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.000000e+00	0.0000000	0.0000000	0.000000e+00	0.0000000
nbr.na	0.000000e+00	0.000000e+00	0.0000000	0.0000000	0.000000e+00	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.000000e+00	0.0000000	0.0000000	0.000000e+00	0.0000000
min	4.000000e+02	8.947260e+00	0.6231900	1.5314000	3.7300e-01	0.0000000	0.0000000	0.0000000	0.2141300	0.2441100	0.4140000	0.1456100	0.7673000	1.9075500	0.1975000	0.1225000	0.0099600	0.1734500	0.7944300	1.729470e+00	0.2444600	0.1320300	4.00940e-01	0.4294500
max	1.611000e+03	4.112035e+01	11.3731300	22.8889900	6.77000e-01	3.3397200	7.6289200	1.0000000	7.1593300	9.3222900	0.9050000	5.0647100	9.7748100	13.9928900	11.1931600	2.7111700	3.5017100	3.9221600	3.8665300	3.012813e+01	4.6936500	1.9345200	6.77050e-01	0.6433400
range	1.211000e+03	3.217309e+01	10.7499400	21.3577000	3.04000e-01	3.3397200	7.6289200	1.0000000	6.9398800	9.0781800	0.4910000	4.9101000	9.0075100	12.9853400	10.9956600	2.5886700	3.4917500	3.7487100	3.0721000	2.820386e+01	4.4401900	1.8024800	2.70110e-01	0.2138900
sum	7.199560e+05	2.521300e+04	4107.6132300	8836.4836200	4.65446e+02	431.0315600	1223.5633900	249.7266000	2675.0835500	3742.3088400	750.7440000	1328.6502900	3157.1498900	4501.2430800	2441.9635500	855.4058800	527.7159100	1597.3990700	2340.6938100	1.072134e+04	1470.0875900	561.0884800	4.89492e+02	531.7586500
median	6.825000e+02	2.510117e+01	3.7755500	8.2204800	4.61000e-01	0.1377800	0.4778350	0.2945000	1.7713200	2.4027950	0.7630000	1.1722250	2.8147000	4.0314500	1.9201700	0.7761050	0.3427050	1.5038450	2.2903850	9.769460e+00	1.3035800	0.5207500	4.86635e-01	0.5295200
mean	7.185190e+02	2.516267e+01	4.0994144	8.8188459	4.64517e-01	0.4301712	1.2211211	0.2492275	2.6709417	2.7368352	0.7492455	1.3259983	3.1508482	4.4922386	2.4370894	0.8536985	0.5266626	1.5942106	2.3360218	1.069994e+01	1.4673533	0.5599685	4.88515e-01	0.5306973
SE.mean	7.180570e+00	1.923830e-01	0.0569771	0.1187133	1.2345e-03	0.0171688	0.0458689	0.0042109	0.0381680	0.0479475	0.0025773	0.0249565	0.0487094	0.0715790	0.0356718	0.0127725	0.0159945	0.0210340	0.0178946	1.517755e-01	0.0230706	0.0070771	9.99800e-04	0.0069928
CI.mean.0.95	1.409970e+01	3.775203e-01	0.1118083	0.2329555	2.42250e-03	0.0336909	0.0900102	0.0082632	0.0748986	0.0940892	0.0050576	0.0489730	0.0955843	0.1404620	0.1092469	0.0250640	0.0313866	0.0412758	0.0351152	2.966572e-01	0.0452841	0.0138877	1.96190e-03	0.0019481
var	5.166370e+04	3.708525e+01	3.2528876	14.1210119	1.52700e-03	0.2955561	2.1081648	0.0177670	1.4597131	2.3035628	0.0066558	0.6240722	2.3773518	5.1337956	3.1955502	0.1634640	0.2563355	0.4433144	0.3208571	2.289973e+01	0.5335943	0.0501859	1.00150e-03	0.0009876
std.dev	2.272965e+02	6.089766e+00	1.8035762	3.7577962	3.90768e-02	0.5434668	1.4519521	0.1332928	1.2081859	1.5177493	0.0815832	0.7899824	1.5418663	2.2657881	1.7622571	0.4043068	0.5062950	0.6658186	0.5664425	4.785367e+00	0.7304754	0.2240221	3.16467e-02	0.0314254
coef.var	3.163403e-01	2.420159e-01	0.4399595	0.4261097	8.41239e-02	1.2633731	1.1893320	0.5348238	0.5833993	0.5545636	0.1088872	0.5957643	0.4893496	0.5043761	0.7230991	0.4735944	0.9613287	0.4176478	0.2424817	4.472329e-01	0.4978862	0.4000620	6.47814e-02	0.0592152

According to the results of the descriptive statistics tables, we can see that the maximum number of games played by the players is 1611, with a maximum number of points scored of 30.12. This is explained by the number of offensive (14) and defensive (97.75) rebounds. Similarly, the maximum number of personal fouls committed by a player or a team is 3.87. Finally, the maximum number of minutes played is 41.12 minutes.

Methodology

In this section, we will demonstrate that the selection of variables is not random but based on a study of the correlation between them. For this reason, we used the technique of principal component analysis (PCA) to reduce the dimension and keep only the relevant and information-rich variables.

Principal component analysis (PCA)

Correlation between variables

Let's examine these variables' correlations to determine which axis to use when constructing and evaluating our PCA. The correlation graph between the different statistical categories will allow us to determine if there are any interesting interactions between them. To do this, we will use the ggcorrplot library to visualize the correlation between each variable. The Pearson correlation coefficient, often abbreviated to r, is a statistical indicator measuring the linear relationship between two data sets. It is also known as bivariate correlation, Pearson product-moment correlation coefficient, or simply correlation coefficient. The symbol ρ often represents the formula used to calculate this measure.Considering a pair of random variables (X, Y), the formula for ρ is :

$$\rho_{X,Y} = \frac{\text{cov(X,Y)}}{\sigma_X \sigma_Y}$$

where: $mboxcov$ is the covariance. $sigma_X$ is the standard deviation of X. $sigma_Y$ is the standard deviation of Y.

We examine the matrix and correlation graphs to see if there are any interesting relationships between the variables.

```
library(ggcorrplot)

# Calculate the correlation matrix
cor_matrix <- cor(nba)
cor_matrix %>%
  kable("latex", booktabs = T,
        caption = "Correlation of NBA Stats.")%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

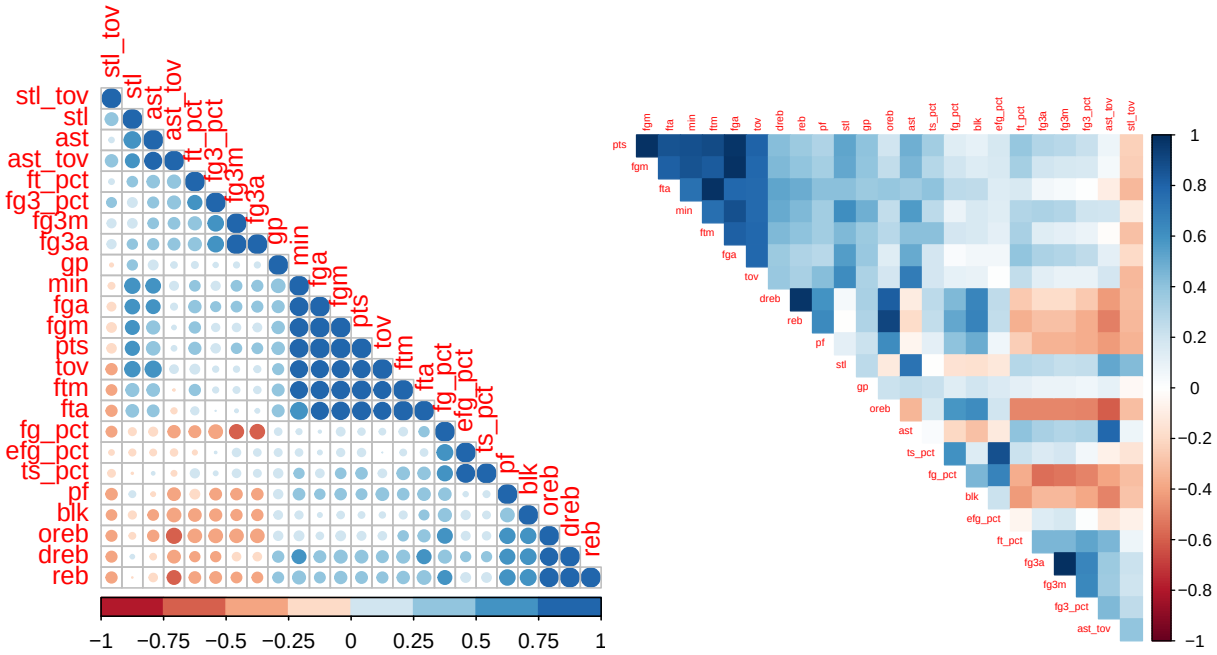
Table 6: Correlation of NBA Stats.

	gp	min	fgm	fga	fg_pct	fg3m	fg3a	fg3_pct	ftm	fta	ft_pct	oreb	dreb	reb	ast	stl	blk	tov	pf	pts	ast_tov	stl_tov	efg_pct	ts_pct
gp	1.0000000	0.4874861	0.3987436	0.3776801	0.1397040	0.0880022	0.0896239	0.0653509	0.3667452	0.3684281	0.1165918	0.2195895	0.3269357	0.3067431	0.2460048	0.2673386	0.1916269	0.3140292	0.2195944	0.4031565	0.1045326	-0.0358093	0.1537354	0.2220208
min	0.4874861	1.0000000	0.8645761	0.8734656	0.0857154	0.2954629	0.3141386	0.2086049	0.7596235	0.7475666	0.2901297	0.2458424	0.5098318	0.4429249	0.5688769	0.6150417	0.1625914	0.7762399	0.3484076	0.8770485	0.1998354	-0.1132655	0.1395972	0.2686977
fgm	0.3987436	0.8645761	1.0000000	0.9817829	0.1931988	0.1781930	0.1937342	0.1683241	0.8358930	0.8191359	0.3326575	0.2425622	0.4407494	0.3951594	0.4558854	0.5119732	0.1401265	0.7868143	0.3362085	0.9850674	0.0492107	-0.2408056	0.1651800	0.2884162
fga	0.3776801	0.8734656	0.9817829	1.0000000	0.0207870	0.2945949	0.3155477	0.2567279	0.8166789	0.7840825	0.3996110	0.1372088	0.3650363	0.3065341	0.5075640	0.5568513	0.0552414	0.7856977	0.2713158	0.9797041	0.1192840	-0.1939131	0.0535514	0.1884321
fg_pct	0.1397040	0.0857154	0.1931988	0.0207870	1.0000000	-0.5328434	-0.5551077	-0.4943863	0.1872252	0.2697747	-0.3672677	0.5877018	0.4452704	0.5104691	-0.2041732	-0.1583210	0.4500167	0.1083948	0.4075293	0.1323865	-0.3957034	-0.3019487	0.6681482	0.6053405
fg3m	0.0880022	0.2954629	0.1781930	0.2945949	-0.5328434	1.0000000	0.9948079	0.6456622	0.1131865	0.0315129	0.4557092	-0.4825970	-0.1824151	-0.2976200	0.2922368	0.2361312	-0.3220105	0.0929365	-0.3419990	0.2764650	0.3531287	0.2096963	0.1787249	0.1829917
fg3a	0.0896239	0.3141386	0.1937342	0.3155477	-0.5551077	0.9948079	1.0000000	0.6508487	0.1326358	0.0522628	0.4520114	-0.4849786	-0.1820780	-0.2985013	0.3218944	0.2693421	-0.3275776	0.1214805	-0.3406081	0.2925005	0.3687983	0.2154731	0.1405372	0.1515791
fg3_pct	0.0653509	0.2086049	0.1683241	0.2567279	-0.4943803	0.6456622	0.6508487	1.0000000	0.0884013	0.0014190	0.5282442	-0.4975992	-0.2698149	-0.3610229	0.3128380	0.2470820	-0.3874222	0.0924957	-0.3603464	0.2225266	0.4449680	0.2506516	0.0124149	0.0517946
ftm	0.3667452	0.7596235	0.8358930	0.8166789	0.1872252	0.1131865	0.1326358	0.0884013	1.0000000	0.9811023	0.3238513	0.2740539	0.4400491	0.4015793	0.4179548	0.4449276	0.1604996	0.7813592	0.3473429	0.8954157	-0.0113736	-0.2913662	0.1109185	0.4178242
fta	0.3684281	0.7475666	0.8191359	0.7840825	0.2697747	0.0315129	0.0522628	0.0014190	0.9811023	1.0000000	0.1647945	0.3864608	0.5237601	0.4982146	0.3591948	0.4127181	0.2519960	0.7778778	0.4144275	0.8683776	-0.0907289	-0.3211679	0.1420558	0.4070014
ft_pct	0.1165918	0.2901297	0.3326575	0.3996110	-0.3672677	0.4557092	0.4520114	0.5282442	0.3238513	0.1647945	1.0000000	-0.4844702	-0.2683724	-0.3514103	0.4099549	0.2821000	-0.4217613	0.2438644	-0.2426471	0.3842718	0.4594105	0.0775817	-0.0582827	0.2154200
oreb	0.2195895	0.2458424	0.2425622	0.1372088	0.5877018	-0.4825970	-0.4849786	-0.4975992	0.2740539	0.3864608	-0.4844702	1.0000000	0.8236670	0.9171640	-0.3310023	-0.1121676	0.6275698	0.1813275	0.6240504	0.1972245	-0.6098955	-0.3096745	0.1999908	0.1746459
dreb	0.3269357	0.5098318	0.4407494	0.3650363	0.4452704	-0.1824151	-0.1820780	-0.2698149	0.4400491	0.5237601	-0.2683724	0.8236670	1.0000000	0.9791952	-0.1073301	0.0472972	0.6547495	0.3706450	0.5991991	0.4226165	-0.4270474	-0.3140030	0.2588524	0.2661282
reb	0.3067431	0.4429249	0.3951594	0.3065341	0.5104691	-0.2976200	-0.2985013	-0.3610229	0.4015793	0.4982146	-0.3514103	0.9171640	0.9791952	1.0000000	-0.1847109	-0.0028915	0.6688832	0.3211651	0.6330403	0.3654550	-0.5039882	-0.3259782	0.2439492	0.2400893
ast	0.2460048	0.5688769	0.4558854	0.5075640	-0.2041732	0.2922368	0.3218944	0.3128380	0.4179548	0.3591948	0.4099549	-0.3310023	-0.1073301	-0.1847109	1.0000000	0.7372684	-0.2950440	0.6953873	-0.0529568	0.4823531	0.7818318	0.0716590	-0.1124091	0.0282332
stl	0.2673386	0.6150417	0.5119732	0.5568513	-0.1583210	0.2361312	0.2693421	0.2470820	0.4449276	0.4127181	0.2821000	-0.1121676	0.0472972	-0.0028915	0.7372684	1.0000000	-0.1632047	0.6285597	0.1148826	0.5250695	0.5123032	0.4343446	-0.1212104	-0.0092636
blk	0.1916269	0.1625914	0.1401265	0.0552414	0.4500167	-0.3220105	-0.3275776	-0.3874222	0.1604996	0.2519960	-0.4217613	0.6275698	0.6547495	0.6688832	-0.2950440	-0.1632047	1.0000000	0.0967675	0.4977177	0.1095776	-0.4954577	-0.2847069	0.2102668	0.1485234
tov	0.3140292	0.7762399	0.7868143	0.7856977	0.1083948	0.0929365	0.1214805	0.0924957	0.7813592	0.7778778	0.2438644	0.1813275	0.3706450	0.3211651	0.6953873	0.6285597	0.0967675	1.0000000	0.4228522	0.8009197	0.1892192	-0.3268233	0.0006478	0.1873119
pf	0.2195944	0.3484076	0.3362085	0.2713158	0.4075293	-0.3419990	-0.3406081	-0.3603464	0.3473429	0.4141275	-0.2426471	0.6240504	0.5991991	0.6330403	-0.0529568	0.1148826	0.4977177	0.4228522	1.0000000	0.3022852	-0.4252576	-0.3465499	0.0650487	0.1160335
pts	0.4031565	0.8770485	0.9850674	0.9797041	0.1323865	0.2764650	0.2925005	0.2225266	0.8954157	0.8687376	0.3842718	0.1972245	0.4226165	0.3654550	0.4823531	0.5250695	0.1095776	0.8009197	0.3022852	1.0000000	0.0743271	-0.2312642	0.1728125	0.3436769
ast_tov	0.1045326	0.1998354	0.0492107	0.1192840	-0.3957034	0.3531287	0.3687983	0.4449680	-0.0113736	-0.0907289	0.4594105	-0.6098955	-0.4270474	-0.5039882	0.7819318	0.5123032	-0.4954577	0.1892192	-0.4252576	0.0743271	1.0000000	0.3999737	-0.1346855	-0.0795197
stl_tov	-0.0358093	-0.1132655	-0.2408056	-0.1939131	-0.3019487	0.2096963	0.2154731	0.2506516	-0.2913662	-0.3211679	0.0775817	-0.3096745	-0.3140030	-0.3259782	0.0716590	0.4343446	-0.2847069	-0.3268233	-0.3465499	-0.2312642	0.3999737	1.0000000	-0.0739973	-0.1486141
efg_pct	0.1537354	0.1395972	0.1651800	0.0535514	0.6681482	0.1787249	0.1405372	0.0124149	0.1109185	0.1420558	-0.0582827	0.1999908	0.2588524	0.2439492	-0.1124091	-0.1212104	0.2102668	0.0006478	0.1892192	0.1728125	-0.1346855	-0.0739973	1.0000000	0.8774660
ts_pct	0.2220208	0.2686977	0.2884162	0.1884321	0.6053405	0.1829917	0.1515791	0.0517946	0.4178242	0.4070014	0.2154200	0.1746459	0.2661282	0.2400893	0.0282332	-0.0092636	0.1485234	0.1873119	0.1160335	0.3436769	-0.0795197	-0.1486141	0.8774660	1.0000000

It can be seen from the correlation table of the NBA statistics that most of the variables are correlated with each other, except for two variables that have a low correlation: `stl_tov` and `gp`.

The `corrplot` package allows plotting a correlation matrix with a confidence interval. It is also very efficient to visualize the details of the correlation between the variables on the graphs we use:

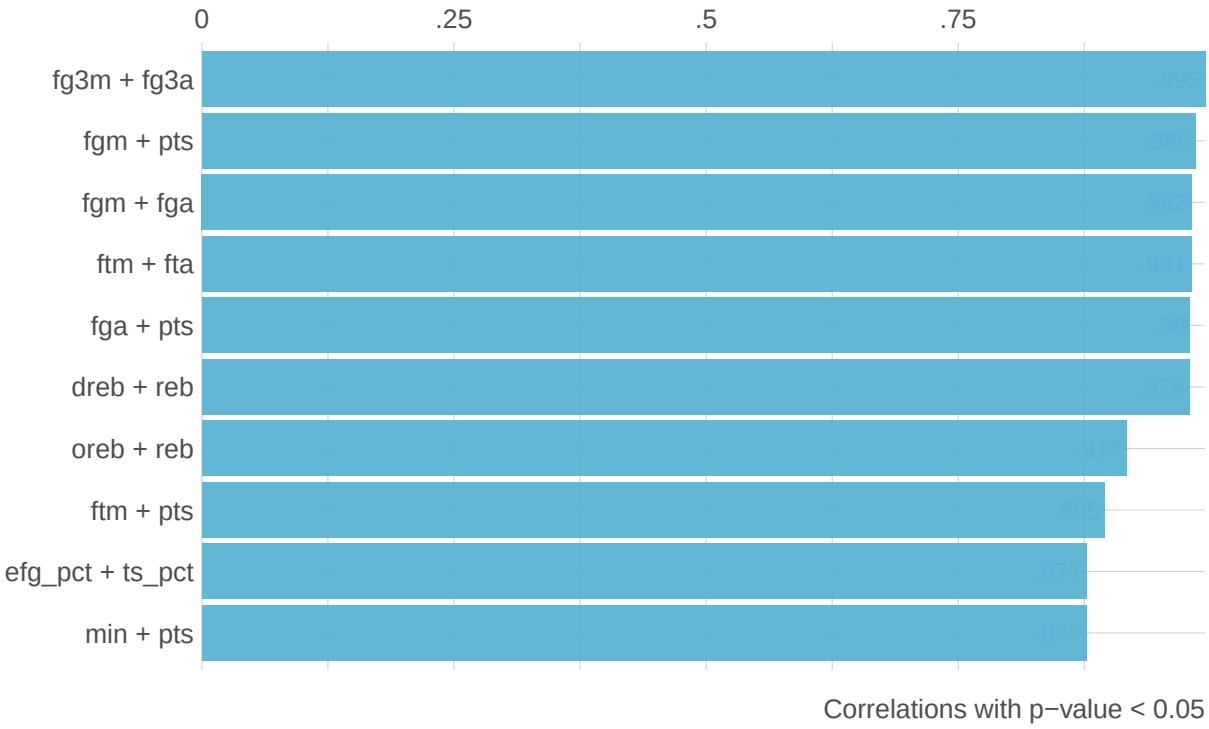
```
library(ggstatsplot)
library(RColorBrewer)
library(corrplot)
library(GGally)
# Create the correlation graph
corrplot(cor_matrix, type = "lower", order = "hclust",
         col = brewer.pal(n = 8, name = "RdBu"))

corrplot(cor_matrix, diag = FALSE, order = "FPC",
         tl.pos = "td", tl.cex = 0.5, method = "color", type = "upper")
```



Note that our input has a dimension $p = 24$, which implies p^{24} different relations between the variables, in total 576 relations, but no way to get information. This is why we need an analytical tool to reduce the dimension. Indeed, let us notice on the graphs that there are strongly correlated variables, especially the most related ones (like `ftm` and `fta`).

We use the `corr_cross()` function to calculate all the correlations and return the highest and most significant ones in a graph:

```
# devtools::install_github("laresbernardo/lares")
library(lares)
Sys.unsetenv("LARES_FONT") # Temporal
corr_cross(nba, # name of the data set
  max_pvalue = 0.05, # have only significant correlations (at the 5% level)
  top = 10 # display the first 10 pairs of variables (by correlation coefficient)
)
```



The cross-correlation plot above shows that the first 10 pairs of variables are positively correlated at the 5% significance level. These pairs are: (`fg3m`, `fg3a`); (`fgm`, `pts`); (`fgm`, `fga`); (`ftm`, `fta`); (`fga`, `pts`); (`dreb`, `reb`); (`oreb`, `reb`); (`ftm`, `pts`); (`efg_pct`, `ts_pct`); (`min`, `pts`). Finally, the exploratory analysis of our data shows that the variables are positively correlated with each other.

Objective of the PCA

The objective of PCA is to simplify a data set by grouping the information contained in many variables into a smaller number of variables. Each variable in a dataset corresponds to a dimension of space, so if the data has three variables, it forms a 3-dimensional space. PCA summarizes the information contained in the variables in the data set by sequentially identifying the directions (or "components") that minimize the average distance between the observations in the data set and that direction. Each direction is a linear grouping of variables. For example, in the case of a 3-variable data set, the direction would be expressed as a $mboxvar_1 + b\ mboxvar_2 + c$ times var_3. Therefore, if one component accurately summarizes the data set by minimizing the mean square distance sufficiently, we can reduce all the data points to a single value by introducing their values for the three variables into the linear combination, making a 3-dimensional point a 1-dimensional projection.

PCA is a technique used to reduce the dimensions of a data set by finding the principal component that minimizes the mean square distance and using a linear combination of the variables to reduce all data points to a single value. If a single component is insufficient, PCA continues to solve the same minimization problem among all components orthogonal to the previous one. This process continues until all the data are captured, which occurs when the number of components equals the number of variables. The objective of PCA is to summarize the data with a minimum of measures.

PCA Analysis of NBA Data

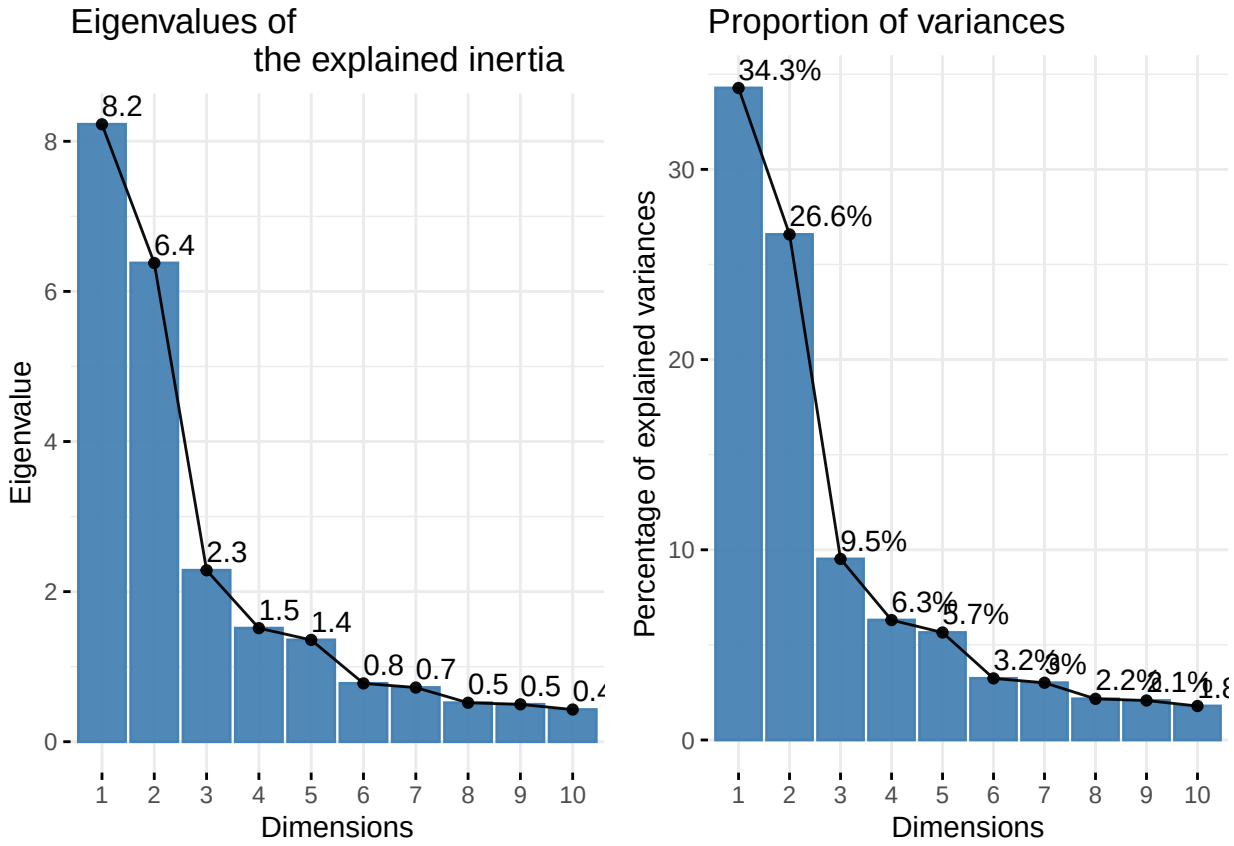
After selecting the numerical variables as raw data, they must be centered and normalized to have identical units for our PCA analysis; we want to use as few components as possible to explain most of the variance. Using too few components may not accurately account for the entire data set (imagine your data points are far away from your first direction: using that one component to summarize your data set would lose a significant amount of information), but using too many components would result in overfitting the data. Here is the amount of information captured by each additional principal component:

```
# Scale/standardize the characteristics to place them on the
#the same scale (all with mean 0, standard deviation = 1)
pca = prcomp(nba, scale=T)
# the same thing, but using SVD instead of clean decomposition if using
#the following function:
# pca = princomp(nba, cor=T)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4    PC5     PC6     PC7
## Standard deviation     2.8682 2.5256 1.51142 1.23009 1.1645 0.88156 0.84972
## Proportion of Variance 0.3428 0.2658 0.09518 0.06305 0.0565 0.03238 0.03008
## Cumulative Proportion  0.3428 0.6085 0.70373 0.76677 0.8233 0.85566 0.88574
##                           PC8     PC9    PC10   PC11   PC12    PC13    PC14
## Standard deviation     0.72053 0.70570 0.65472 0.6500 0.6060 0.41614 0.30767
## Proportion of Variance 0.02163 0.02075 0.01786 0.0176 0.0153 0.00722 0.00394
## Cumulative Proportion  0.90737 0.92812 0.94598 0.9636 0.9789 0.98611 0.99005
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.30271 0.23671 0.19467 0.17058 0.11126 0.06634 0.05948
## Proportion of Variance 0.00382 0.00233 0.00158 0.00121 0.00052 0.00018 0.00015
## Cumulative Proportion  0.99387 0.99620 0.99778 0.99899 0.99951 0.99969 0.99984
##                           PC22    PC23      PC24
## Standard deviation     0.04715 0.03983 1.207e-06
## Proportion of Variance 0.00009 0.00007 0.000e+00
## Cumulative Proportion  0.99993 1.00000 1.000e+00
```

Eigenvalues and eigenvectors are two different concepts in PCA analysis. Eigenvalues are numbers that indicate the importance of each principal component, while eigenvectors are vectors that describe the direction of the principal components. Eigenvalues are often expressed as a percentage to indicate the proportion of total variance explained by each principal component. The contributions of the variables to the principal components are calculated from the eigenvectors and can be used to interpret the contribution of each variable to each principal component.

```
p1_ess <- fviz_eig(pca, addlabels = TRUE, choice='eigenvalue',main ="Eigenvalues of
                  the explained inertia")
p2_ess <- fviz_eig(pca, addlabels = TRUE, main="Proportion of variances")
grid.arrange(p1_ess, p2_ess, nrow=1,name = "arrange")
```



Three components explain 70.37% of the data, and another fourth direction defines another 6.3%. Since the fourth component does not add much to our understanding, we will use the first three components to transform our 24-dimensional data set into a 3-dimensional one. Otherwise, we extracted the eigenvalues on the principal components (see the plot above left). We then visualised the percentage of variances explained (i.e., the information) of these eigenvalues in 10 dimensions (see the plot on the right). Therefore, we can project the data into these dimensions.

Note that the sum of the squares of the factors (eigenvectors) is equal to 1 because they are normalised vectors (module 1).

```
sum(pca$rotation[,1]^2)
```

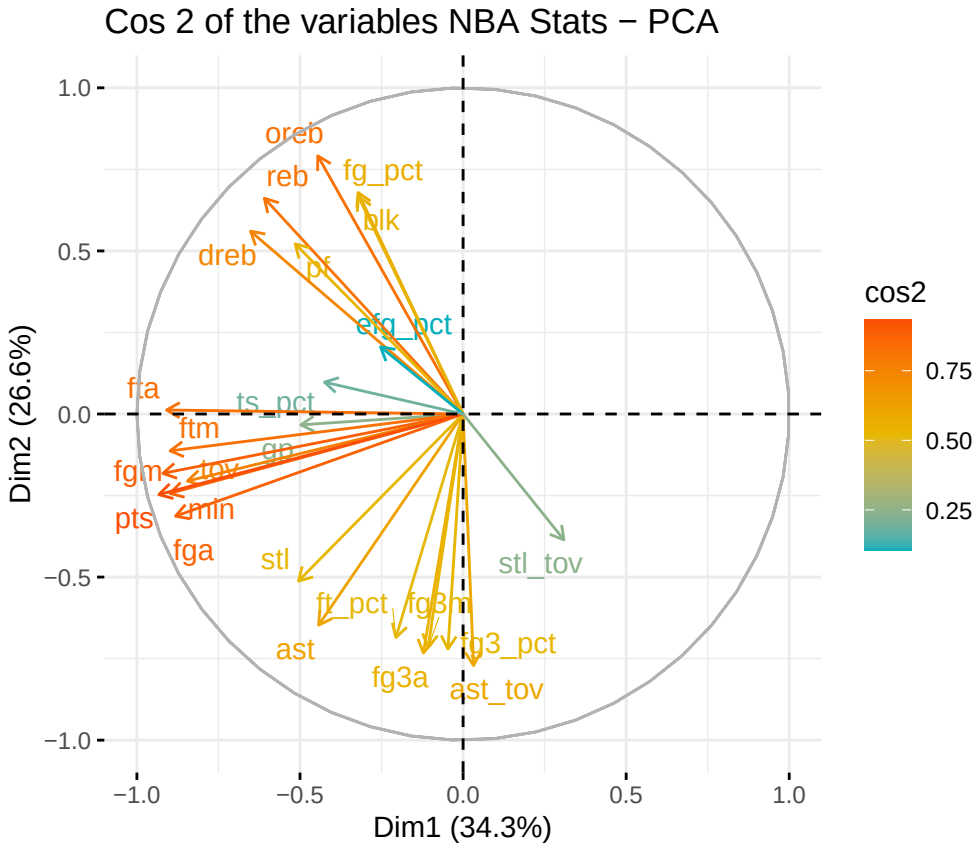
```
## [1] 1
```

This means that the eigenvectors are more accessible to interpret than the eigenvalues. They are like percentages (numbers between 0 and 1). In the following section, we will plot these eigenvectors. They are called the contribution of the variables to the components.

Quality of the representation

The dimensions of the multivariate data are reduced through PCA, which can be represented visually with little loss of information. The three graphs below will be studied more thoroughly as they provide complete information on particular variables. To do this, we use the function `fviz_pca_var` :

```
fviz_pca_var(pca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid overlapping texts
             ,title= "Cos 2 of the variables NBA Stats - PCA"
             )
```



On the projection plane of the PCA of the variables above, we can observe the relationships between the statistical variables of the NBA players. The two dimensions represented on the graph explain 34.3% and 26.6% of the total variance of the data, respectively. It is clear that the variables `min`, `pts`, `fgm`, `fga`, `tov`, `ast`, `ftm`, and `fta` are negatively correlated with the first dimension, while the variables `oreb`, `dreb`, `reb`, and `fg_pct` are positively correlated with the second dimension. The distances of the variables from the original measure the degree of representation of these variables on the principal axes. Variables positively correlated with a high cos2 indicate a good representation of these axes. In contrast, variables negatively correlated with a low cos2 indicate that the principal axes do not perfectly represent the variable.

Contributions of the variables to the principal axes

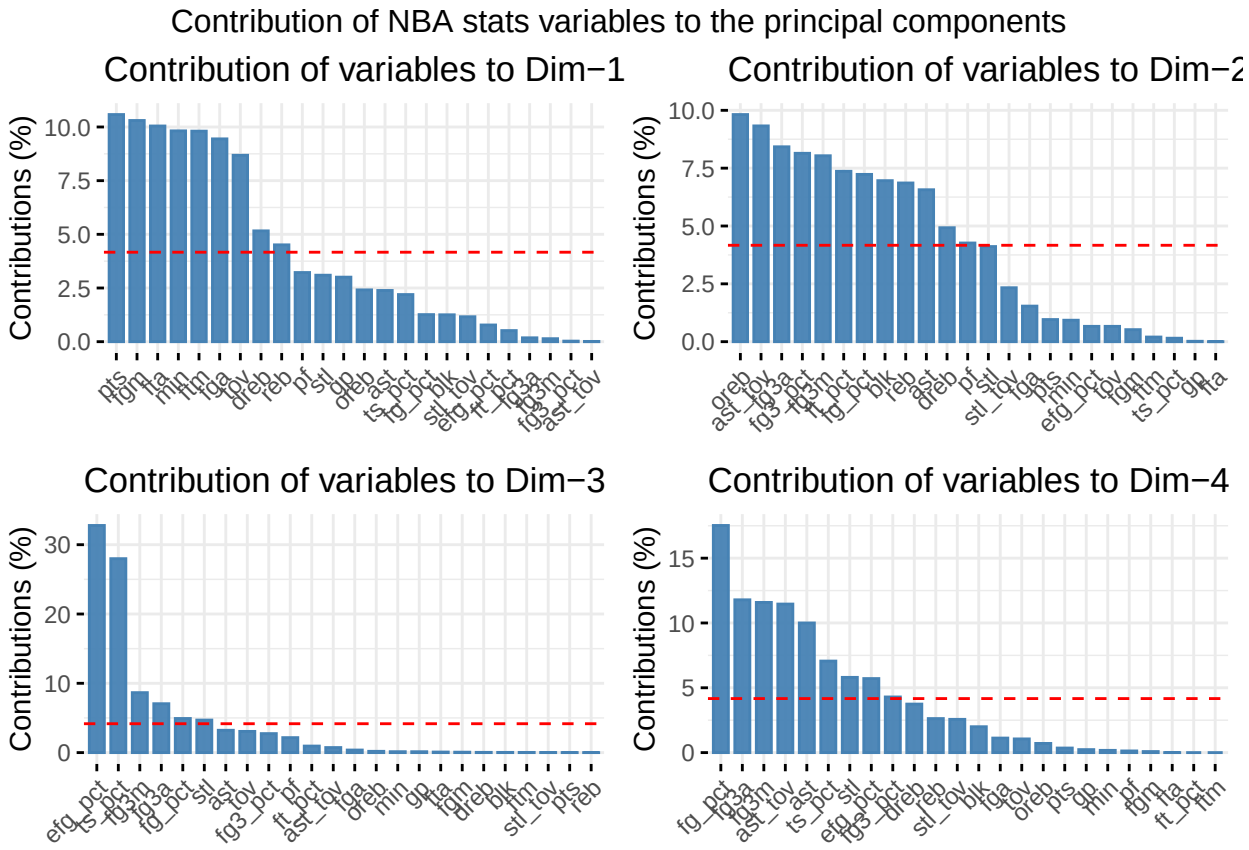
The contributions of the variables in a specific principal axis are expressed as a percentage. The larger the value of the contribution, the more the variable contributes to the principal component in question.

- Variables correlated with PC1 (i.e., Dim.1) and PC2 (i.e., Dim.2) are the most important in explaining the variability in the data set.
- Variables that are not correlated with any axis or that are correlated with the last few axes are low-contribution variables and can be removed to simplify the analysis in question.

To create a bar graph of the contribution of variables, the function `fviz_contrib()` can be used as follows:

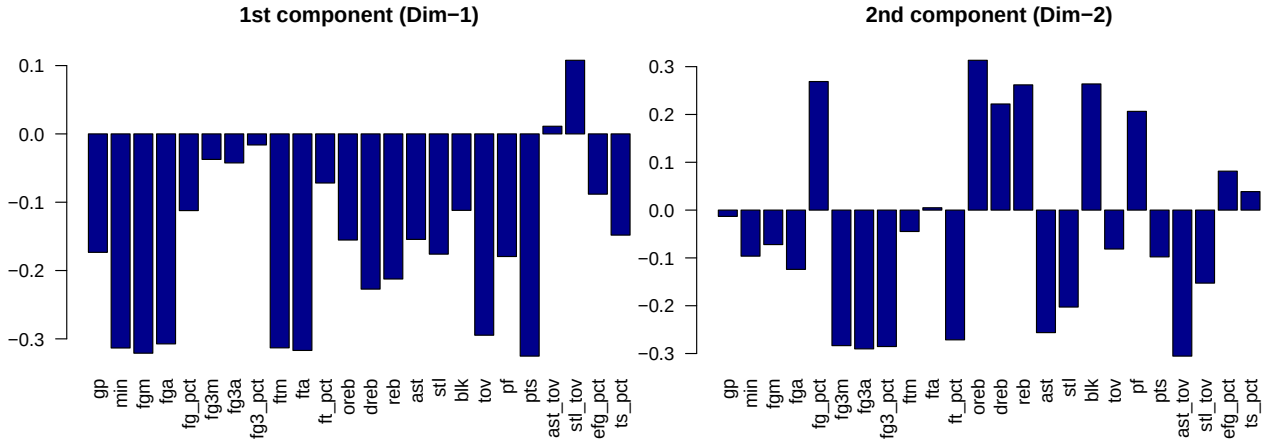
```
# Contributions of variables to PC1
c1 <- fviz_contrib(pca, "var", axes=1)
# Contributions of variables to PC2
c2 <- fviz_contrib(pca, "var", axes=2)
# Contributions of variables to PC3
c3 <- fviz_contrib(pca, "var", axes=3)
# Contributions of variables to PC4
```

```
c4 <- fviz_contrib(pca, "var", axes=4)
# Combine the contributions of the variables to PC1, PC2, PC3, and PC4.
grid.arrange(c1, c2, c3, c4, top='Contribution of NBA stats variables to the principal components')
```

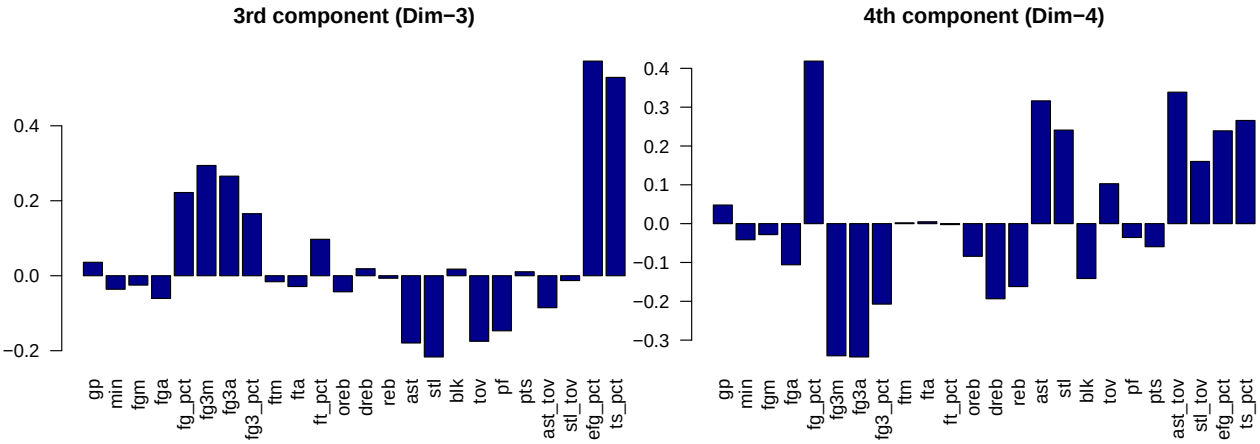


We can visualize the 4 principal components using `barplot`:

```
barplot(pca$rotation[,1], las=2, col="darkblue", main = "1st component (Dim-1)")
barplot(pca$rotation[,2], las=2, col="darkblue", main = "2nd component (Dim-2)")
```




```
barplot(pca$rotation[,3], las=2, col="darkblue", main = "3rd component (Dim-3)")
barplot(pca$rotation[,4], las=2, col="darkblue", main = "4th component (Dim-4)")
```



The results show that the variables `pts`, `fgm`, `fta`, `min`, `ftm`, `fga` are the main contributors in terms of the first factor. In terms of the second factor, the variables included are mainly `oreb`, `ast_tov`, `fg3a`, `fg3_pct`, `fg3m`, `ft_pct`. The third factor is only designated by variables `efg_pct`, `ts_pct`. Finally, the variables contibutor more of the last component are: `fg_pct`, `fg3a`, `fg3m`, `ast_tov`. We can also see that the variables `fg3m`, `fg3a`, `fg_pct`, contribute most to dimensions 2, 3, and 4. The red dotted line in the above graph indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be $refrac{1}{mbox{length(variables)}} = $refrac{1}{24} = 4.16%.

Contribution of individuals in the PCA

We can now rank the players according to their first PC score: the best historical players in terms of performance :

```
calculateScore = function(data) {
  return(sum((pca$rotation[, 1]*data)^2))
}
historical_players.df$player_name[sort.int(apply(nba, 1, calculateScore), decreasing = T,
                                           index.return = T)$ix[1:10]]
```

```
##  [1] "Robert Parish"       "Kareem Abdul-Jabbar" "John Stockton"
##  [4] "Karl Malone"         "Kevin Garnett"       "Kevin Willis"
##  [7] "Dirk Nowitzki"       "Tim Duncan"          "Jason Kidd"
## [10] "Reggie Miller"
```

Perhaps we can get more information by ranking the players using this component:

```
calculateScore = function(data) {
  return(sum((pca$rotation[, 2]*data)^2))
}
nba$gp[sort.int(apply(nba, 1, calculateScore), decreasing = T, index.return = T)$ix[1:10]]
```

```
##  [1] 1611 1560 1504 1476 1462 1424 1392 1394 1391 1389
```

Let's look at the contribution of each player to the components. For the $i - mboxth$ player and the first component, the contribution is $z_{1,i}^2/\lambda_1/n$, a number between 0 and 1.

```
head(get_pca_ind(pca)$contrib[,1]) # This is in %, i.e. between 0 and 100
```

```
##         1         2         3         4         5         6
## 1.0588322 0.8509603 0.9762763 0.7415180 0.5860088 1.0777332
```

```
head((pca$x[,1]^2)/(pca$sdev[1]^2))/dim(nba)[1] # This number is between 0 and 1
```

```
## [1] 0.010588322 0.008509603 0.009762763 0.007415180 0.005860088 0.010777332
```

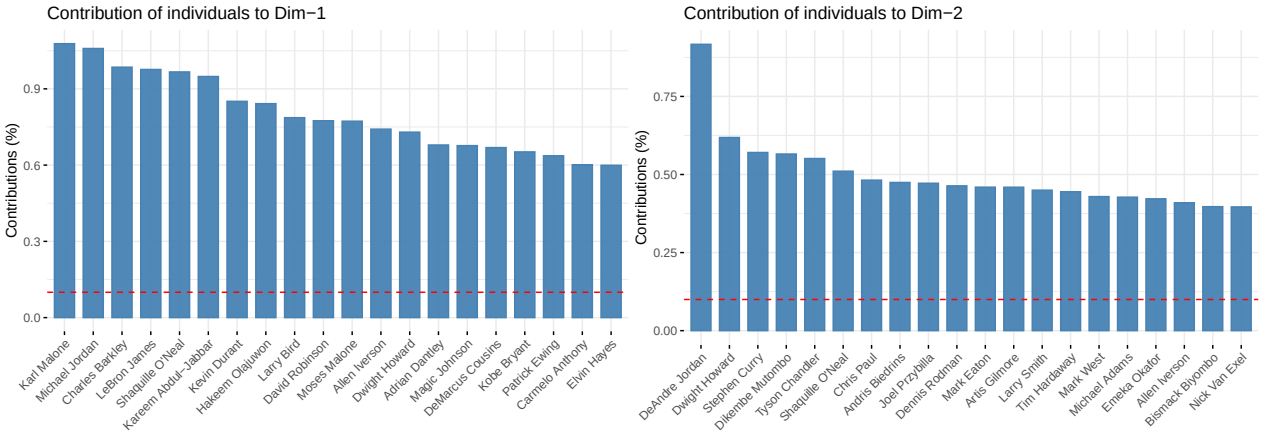
We will view the contributions of all players to the first component, let's see the names of each player:

```
#  It is similar to names[order(pca$x[,1])][1:10] but in percentage
names[order(get_pca_ind(pca)$contrib[,1],decreasing=T)][1:10]
```

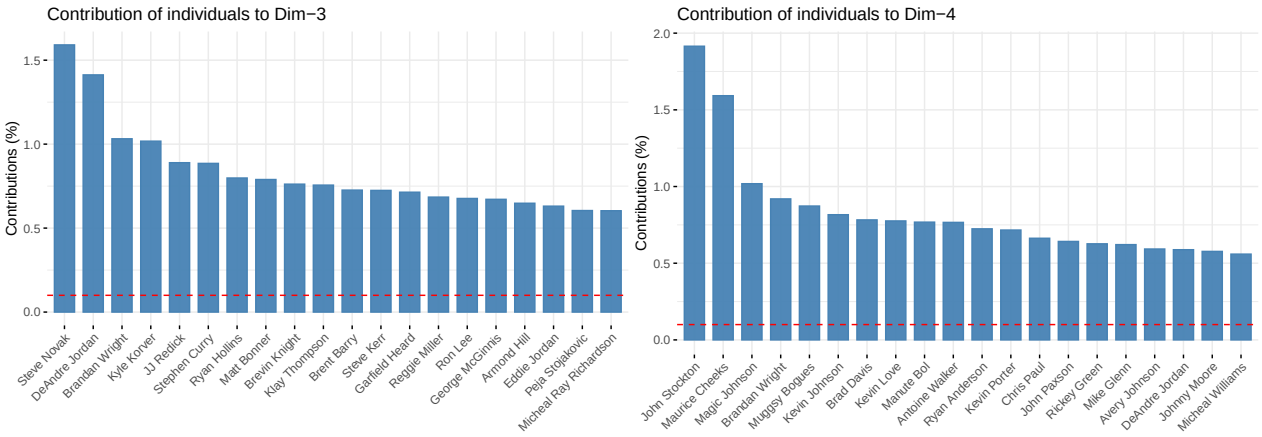
```
##  [1] "Karl Malone"         "Michael Jordan"      "Charles Barkley"
##  [4] "LeBron James"        "Shaquille O'Neal"    "Kareem Abdul-Jabbar"
##  [7] "Kevin Durant"        "Hakeem Olajuwon"     "Larry Bird"
## [10] "David Robinson"
```

Finally, to see the top 20 players in terms of contributions on the 4 main components:

```
names_z1 = names[order(get_pca_ind(pca)$contrib[,1],decreasing=T)]
fviz_contrib(pca, choice = "ind", axes = 1, top=20)+scale_x_discrete(labels=names_z1)
names_z2 = names[order(get_pca_ind(pca)$contrib[,2],decreasing=T)]
fviz_contrib(pca, choice = "ind", axes = 2, top=20)+scale_x_discrete(labels=names_z2)
```




```
names_z3 = names[order(get_pca_ind(pca)$contrib[,3],decreasing=T)]
fviz_contrib(pca, choice = "ind", axes = 3, top=20)+scale_x_discrete(labels=names_z3)
names_z4 = names[order(get_pca_ind(pca)$contrib[,4],decreasing=T)]
fviz_contrib(pca, choice = "ind", axes = 4, top=20)+scale_x_discrete(labels=names_z4)
```



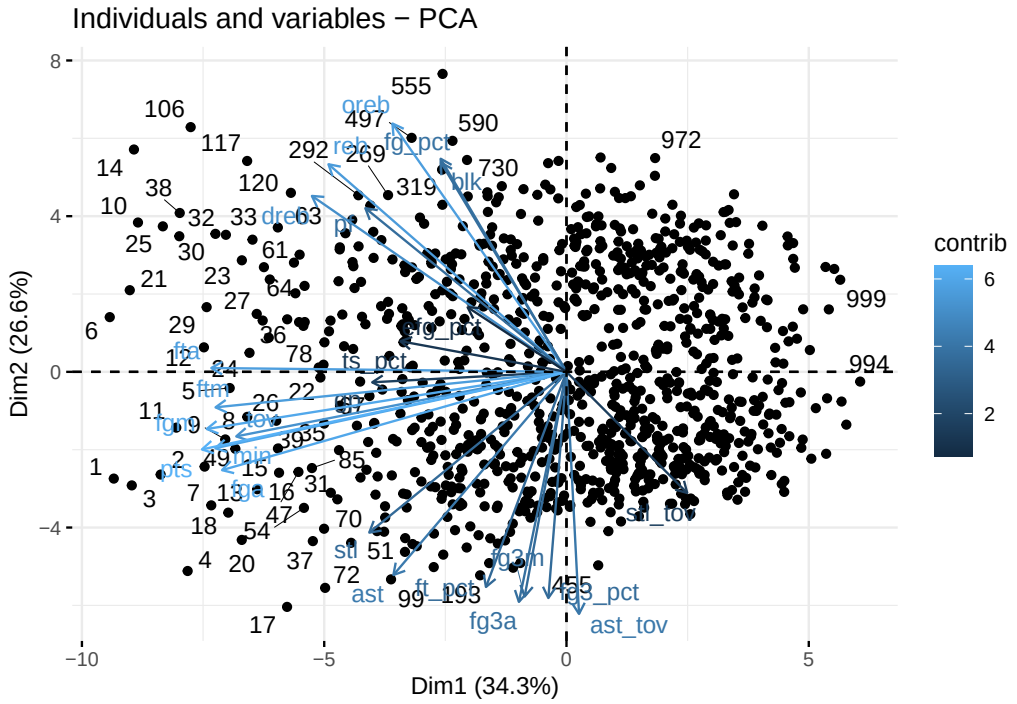
We find that all of the 20 selected players contributed strongly to the four main components of our study. In particular, the players who contributed the most were Karl Malone, Michael Jordan, Dwight Howard, DeAndre Jordan, Steve Novak, John Stockton and Maurice Cheeks. Thus, based on these results, we can say that the best NBA players of all time are those with a wide variety of basketball skills.

Projection of observations and variables

We will use the Biplot to project the observations and the variables in the same plane (using the first two components). For an elegant visualisation of the individuals and variables on the PCA plane, we will use the following function `fviz_pca_biplot`:

```
fviz_pca_biplot(pca, repel = TRUE,col.var = "contrib", title = "Individuals and variables - PCA")
```

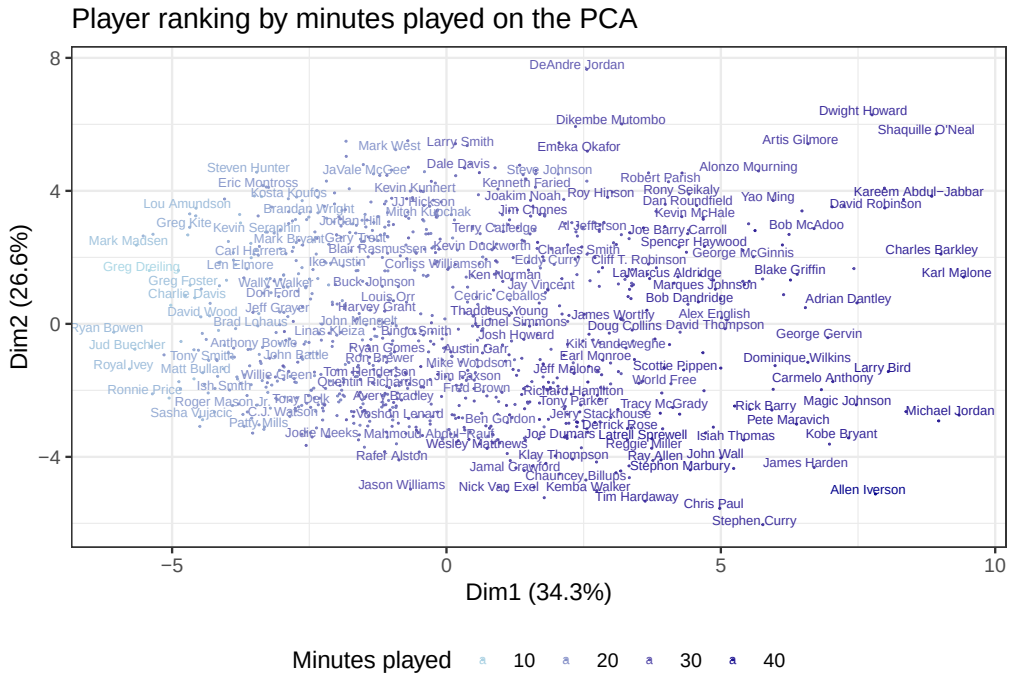
```
## Warning: ggrepel: 938 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



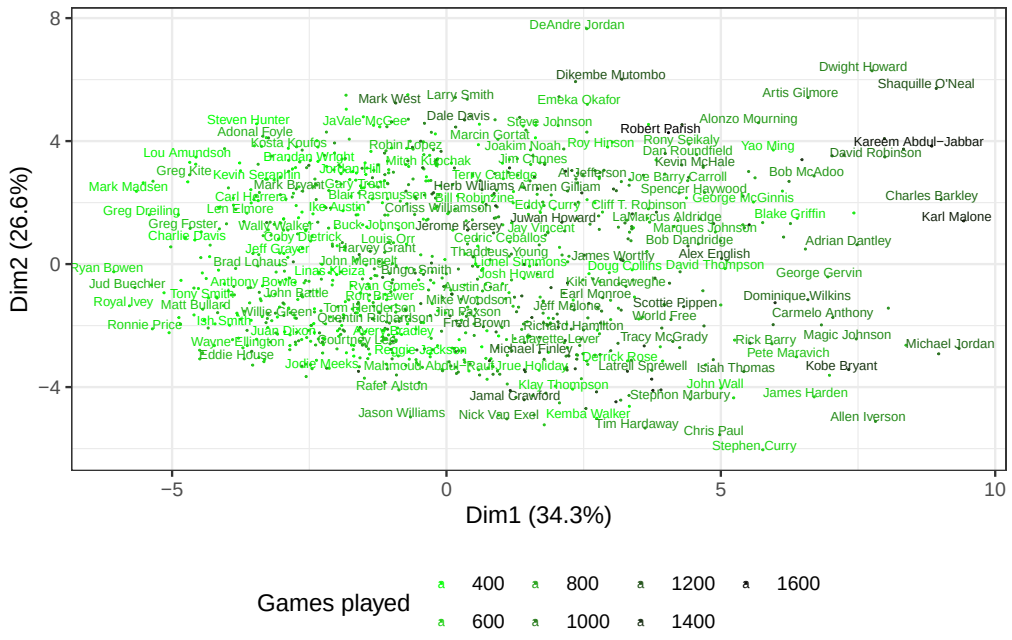
PCA Scores: Player Projection

Recall that for the $j-$mbox{th}$ principal component: $Z_j = Xa_j$, a_j represents the eigenvectors, and Z_j represents the scores or factors of each principal component. We will plot the first two scores (i.e. PC1, PC2), using colors for minutes played (`min`) and games played (`gp`) :

```
data.frame(z1=-pca$x[,1],z2=pca$x[,2]) %>%
  ggplot(aes(z1,z2,label=names, ,color=nba$min)) + geom_point(size=0) +
  labs(title="Player ranking by minutes played on the PCA", x="Dim1 (34.3%)",
       y="Dim2 (26.6%)") +
  theme_bw() + scale_color_gradient(low="lightblue", high="darkblue")+
  theme(legend.position="bottom") + geom_text(size=2, hjust=0.6, vjust=0,
                                              check_overlap = TRUE)+
  guides(color = guide_legend(title = "Minutes played"))
```




```
data.frame(z1=-pca$x[,1],z2=pca$x[,2]) %>%
  ggplot(aes(z1,z2,label=names,color=nba$gp)) + geom_point(size=0) +
  labs(title="Ranking of players per game played on the PCA",x="Dim1 (34.3%)",
       y="Dim2 (26.6%)") +
  theme_bw() + scale_color_gradient(low="green", high="black")+
  theme(legend.position="bottom") + geom_text(size=2, hjust=0.6, vjust=0,
                                              check_overlap = TRUE)+
  guides(color = guide_legend(title = "Games played"))
```



We can see here the distribution between the players. The best NBA players have a very high PC1 score and are on the right. The forwards have a very high PC2 and are on the top. There are

also differences and similarities based on the position between players based on overall statistics. For example, Magic Johnson and Michael Jordan are close to each other, indicating that famous players can be grouped. So we can rank any player based on these two PCA graphs and make distinctions. The ones on the top right are the best offences, such as; DeAndre Jordan, Dwight Howard, Shaquille O'Neal, Artis Gilmore, Dikembe Mutombo, Moses Malone, Alonzo Maurning, Kareem Abdyl-Jabbar... etc. And the ones on the bottom right are the best defences like Stephen Curry, Chris Paul, Alen Iverson, Gilbert ArenasJames Harden, Kobe Bryant, Russell Westbrook, Michael Jordan... etc.

Clustering

In the second part of this project, we will use the two clustering methods, namely hierarchical clustering and K-means, on our NBA player performance database to answer the second question: how to help NBA coaches make more relevant decisions and evaluate whether player choices are the right ones. Both methods are unsupervised learning because we are looking for a new set of labels (clusters) for our data. It would be possible to group players into different groups based on their performance, such as players who played more minutes and games, were influential on defence, etc. This could be useful for statistical analysis and decision-making in-game strategy and team composition.

Clustering hierarchical

Hierarchical clustering is a method of grouping data based on the proximity of elements to each other. It creates a data clustering tree, called a dendrogram, where each node represents a group of data and each branch represents the proximity between these groups. The process we will follow for this method is as follows:

- We will calculate the Euclidean distance matrix for the variables in our dataset.
- We will attempt hierarchical clustering using the nearest neighbour and centroid clustering methods and then Ward's minimum variance method.

The first clustering method we will try is hierarchical clustering. Dendrograms (tree diagrams) can help visualise the separation of NBA players. Clusters of players are determined using the distances between observations, trying to maximise the space between groups and minimise the distance within groups. The most commonly used distance formula is the Euclidean distance:

$$\text{distance(x,y)} = \sqrt{\sum_{i}^{n}(x_i - y_i)}$$

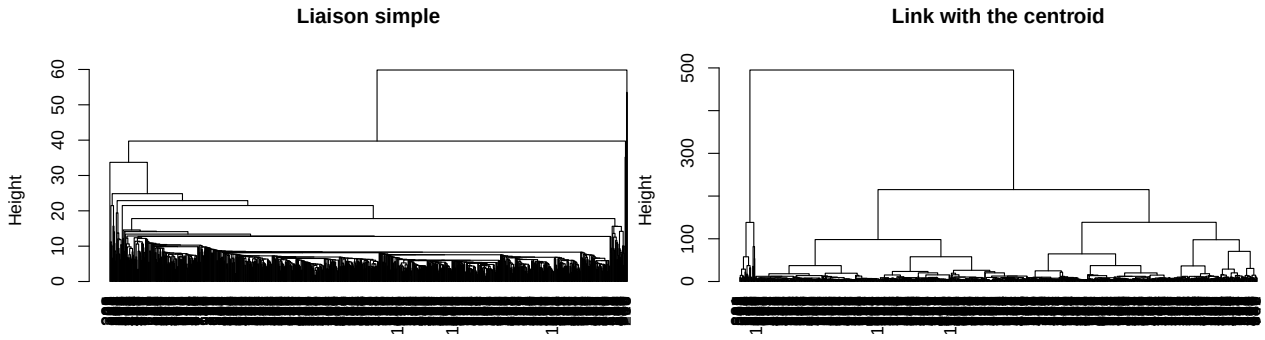
We work in the quantitative statistical variables of the NBA using the z-score formula for the normal distribution:

$$x_{new} = \frac{x_i - \bar{x}}{\sigma_x}$$

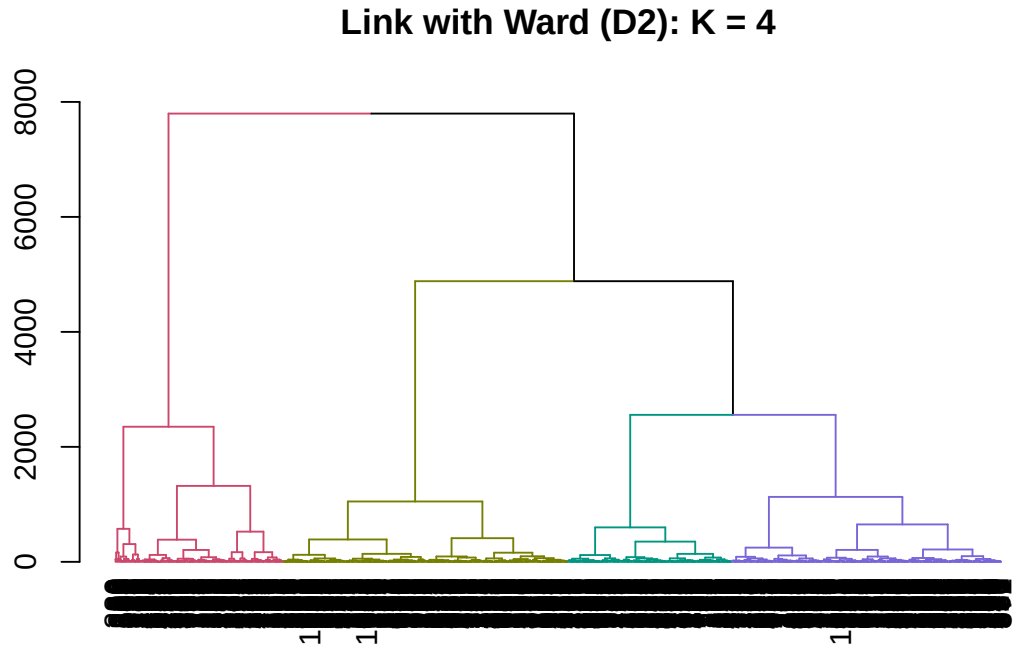
```
# Compute the Euclidean distance matrix for the features :
nba_dist <- dist(nba, method = 'euclidean')

# Try clustering by single link, centroid, and ward (D2):
hcl_single <- hclust(d = nba_dist, method = 'single')
hcl_centroid <- hclust(d = nba_dist, method = 'centroid')
hcl_ward <-  hclust(d = nba_dist, method = 'ward.D2')
```

```
# Nearest neighbor method:
plot(hcl_single, hang = -1, main = 'Liaison simple', xlab = '', sub = '')
```

```
# Centroid groups :
plot(hcl_centroid, hang = -1, main = 'Link with the centroid', xlab = '',  sub = '')
```




```
# Ward's minimum variance method,
# with squared dissimilarities before clustering :
dend <- as.dendrogram(hcl_ward)
hcl_k <- 4
dend_col <- color_branches(dend, k = hcl_k)
plot(dend_col, main = paste0('Link with Ward (D2): K = ', hcl_k))
```



Ward's dendrogram appears to be the best of the three methods because of the clear separation between clusters. The simple linkage method and the centroid linkage method do not provide a clear separation of sets and produce many sparsely populated clusters. Since Ward's dendrogram is the best of the three, we will study its distribution for three and four clusters, providing the best separation level.

Let's look at the distribution of players in each cluster for the solutions $k = 3$ and $k = 4$:

```
# Add the cluster assignments for the solutions k = 3 and k = 4 for the model
# Hierarchical Clustering method to the main data set:
historical_players.df$hcl_ward_labs_three <- cutree(hcl_ward, k = 3)
historical_players.df$hcl_ward_labs_four <- cutree(hcl_ward, k = 4)

# Organize in a data frame:
```

```
hcl_df_three <- data.frame(table(historical_players.df$hcl_ward_labs_three))
hcl_df_four <- data.frame(table(historical_players.df$hcl_ward_labs_four))
col_names <- c('Clusters', 'Count')
colnames(hcl_df_three) <- col_names
colnames(hcl_df_four) <- col_names

hcl_df_three[4,] <- NA
df_clusters <- data.table(hcl_df_four$Clusters, hcl_df_three$Count, hcl_df_four$Count)

#Change column names
df_clusters <- df_clusters %>%
    dplyr::rename(Clusters = V1, 'k = 3' = V2, 'k = 4' = V3)
```

```
df_clusters %>%
  kable("latex", booktabs = T,
        caption = "The clusters of the hierarchical clustering method")%>%
  kable_styling(latex_options = c("striped","HOLD_position"),font_size = 20)
```

Table 7: The clusters of the hierarchical clustering method

Clusters	k = 3	k = 4
1	190	190
2	490	305
3	322	185
4	NA	322

The distribution of players between clusters is uneven in both solutions, with the fewest players in the 3rd and 4th clusters for $k = 3$ and $k = 4$ clusters, respectively.

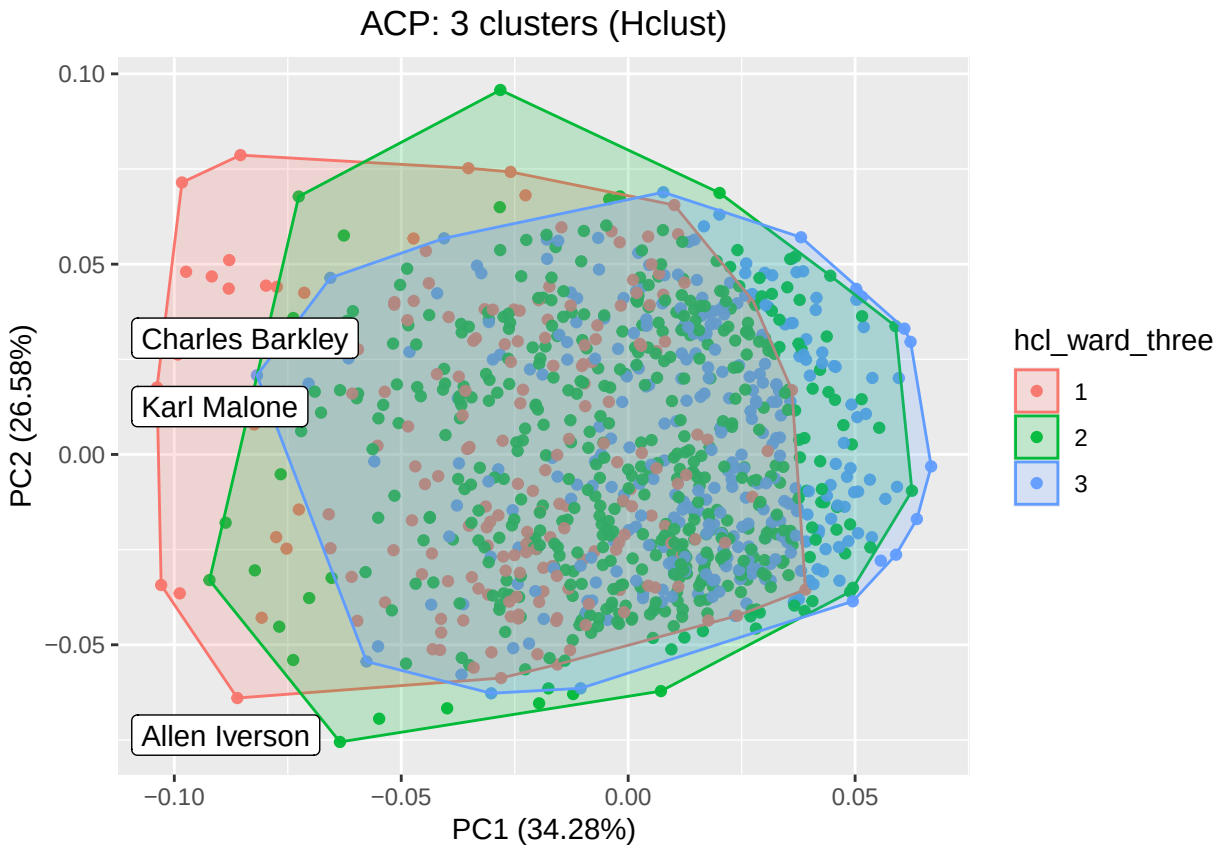
We will create a function to plot the PCA and label the points:

```
plot_pca <- function(object, frame = FALSE, x = 1, y = 2,
                     data, colour, title, label) {#, leg_title) {
  #  plots the data in the PCA space
  # object = PCA or K-Means object
  # x = which PC for x-axis (1, 2, ,3, etc..)
  # y = which PC for y-axis (1, 2, 3, etc..)
  # data = underlying data
  p <- autoplot(pca, x = x, y = y, data = historical_players.df, colour = colour,
                frame = frame) +
        ggtitle(title) +
        # center title
        theme(plot.title = element_text(hjust = 0.5)) +
        geom_label_repel(aes(label = label),
                         box.padding   = 0.35,
                         point.padding = 0.5,
                         segment.color = 'grey50')
  return(p)
```

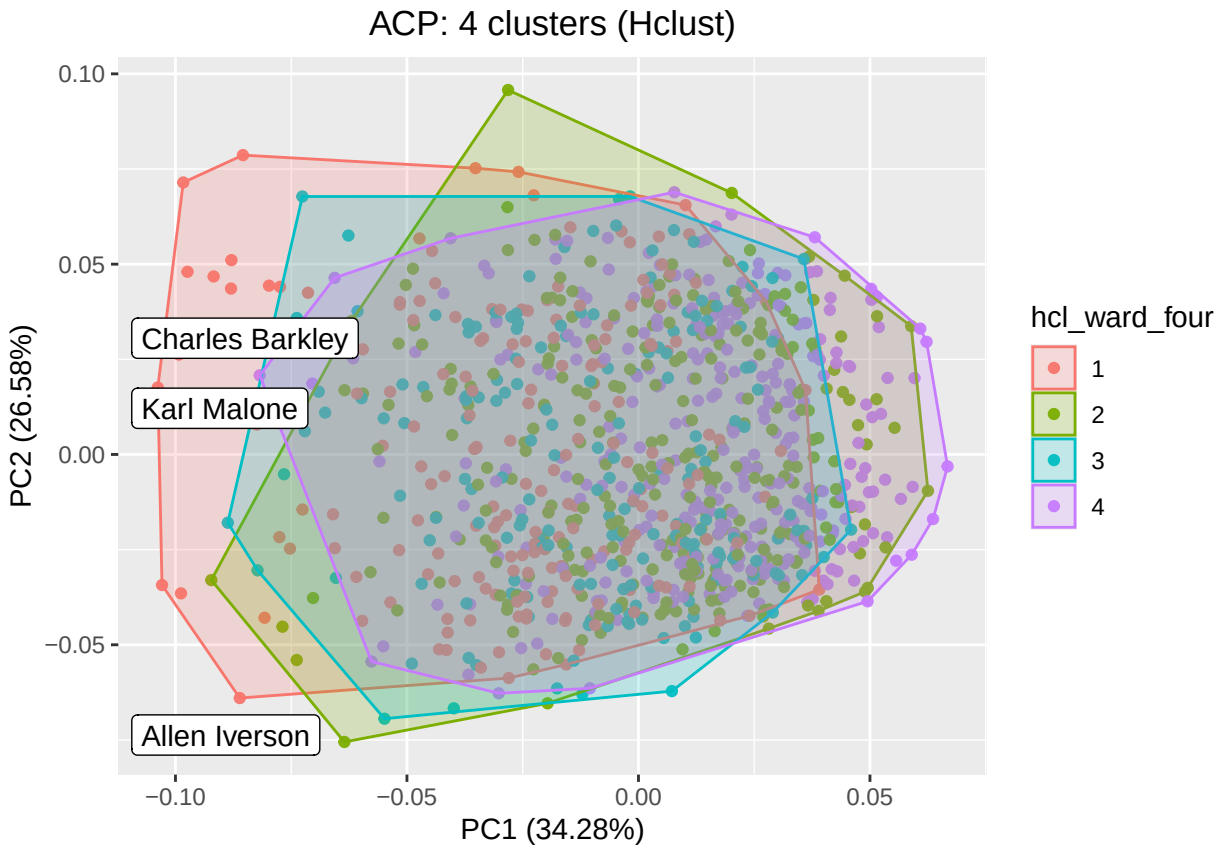
```
}
```

```
# Factor the labels into the data set:
historical_players.df$hcl_ward_three <- factor(cutree(hcl_ward, k = 3))
historical_players.df$hcl_ward_four <- factor(cutree(hcl_ward, k = 4))

# Include some player names in the graph:
hcl_labels <- ifelse(historical_players.df$min >= 36 | historical_players.df$min <= 2.5 |
                     (historical_players.df$min >= 28.8 & historical_players.df$min <= 29) |
                     (historical_players.df$min >= 25 & historical_players.df$min <= 25.2),
                     as.character(historical_players.df$player_name), '' )

# elements to be browsed in a loop
hcl_labs <- names(historical_players.df %>% select(tail(names(.), 2)))
hcl_ks <- c(3, 4)

# Plot the hclust labels overlaid on the PCA
for (i in seq_along(hcl_labs)) {
  p <- plot_pca(pca, frame = TRUE,
                data = historical_players.df, colour = hcl_labs[i],
                title = paste0('ACP: ', hcl_ks[i], ' clusters (Hclust)'),
                label = hcl_labels
  )

  print(p)
}
```

```
## Warning: ggrepel: 48 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
## Warning: ggrepel: 48 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



When viewing the clusters in PC space, we can see a clear separation between the 3- and 4-cluster solutions, with some overlapping points. The hierarchical clustering graphs for $k = 3$ and $k = 4$ indicate the types of players to consider more closely to help the coach make the right decision. The ideal players for the NBA team would be Allen Iverson, Karl Malone, Charles Barkley, and Kareem Abdul-Jabbar, who play between 36 minutes and 2.5 minutes, between 28.8 minutes and 29 minutes, and between 25 minutes and 25.2 minutes, respectively, and who often start games and score many points.

K-Means Clustering

K-means is a clustering method based on the distribution of data around k centers of gravity. The data are assigned to the group whose center of gravity is the closest. This method is efficient for data with a clearly defined global structure and requires to know in advance the desired number of groups (k). In our case, we explored the use of K-Means for clustering with a 4 cluster solution based on the Silhouette index to evaluate the separation between clusters. The graphs below show:

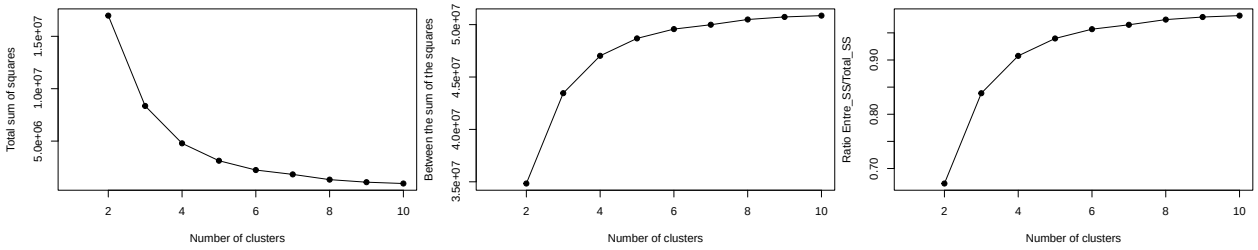
1) The Silhouette index for the K-Means method.
2) Cluster assignments in the PC space.
3) The distribution of players between clusters.

The following three functions allow to visualize the ratio between the total sum of squares within (`wss`), the sum of squares between (`bss`) and the total sum of squares (`totss`) :

```
wss <- function(data, maxCluster = 10) {
  # Initialize the sum of squares
  set.seed(50)
  SSw <- (nrow(data) - 1) * sum(apply(data, 2, var))
  SSw <- vector()
  for (i in 2:maxCluster) {
```

```
    SSw[i] <- sum(kmeans(data, centers = i)$tot.withinss)
  }
  plot(1:maxCluster, SSw, type = "o", xlab = "Number of clusters",
       ylab = "Total sum of squares", pch=19)
}

bss <- function(data, maxCluster = 10) {
  #Initialize the sum of squares
  set.seed(50)
  SSw <- (nrow(data) - 1) * sum(apply(data, 2, var))
  SSw <- vector()
  for (i in 2:maxCluster) {
    SSw[i] <- sum(kmeans(data, centers = i)$betweenss)
  }
  plot(1:maxCluster, SSw, type = "o", xlab = "Number of clusters",
       ylab = "Between the sum of the squares", pch=19)
}

btratio <- function(data, maxCluster = 10) {
  # Initialize the sum of squares
  set.seed(50)
  SSw <- (nrow(data) - 1) * sum(apply(data, 2, var))
  SSw <- vector()
  for (i in 2:maxCluster) {
    SSw[i] <- sum(kmeans(data, centers = i)$betweenss/kmeans(data, centers = i)$totss)
  }
  plot(1:maxCluster, SSw, type = "o", xlab = "Number of clusters",
       ylab = "Ratio Entre_SS/Total_SS", pch=19)
}
```

```
# obtaining k optimum
wss(nba)
bss(nba)
btratio(nba)
```

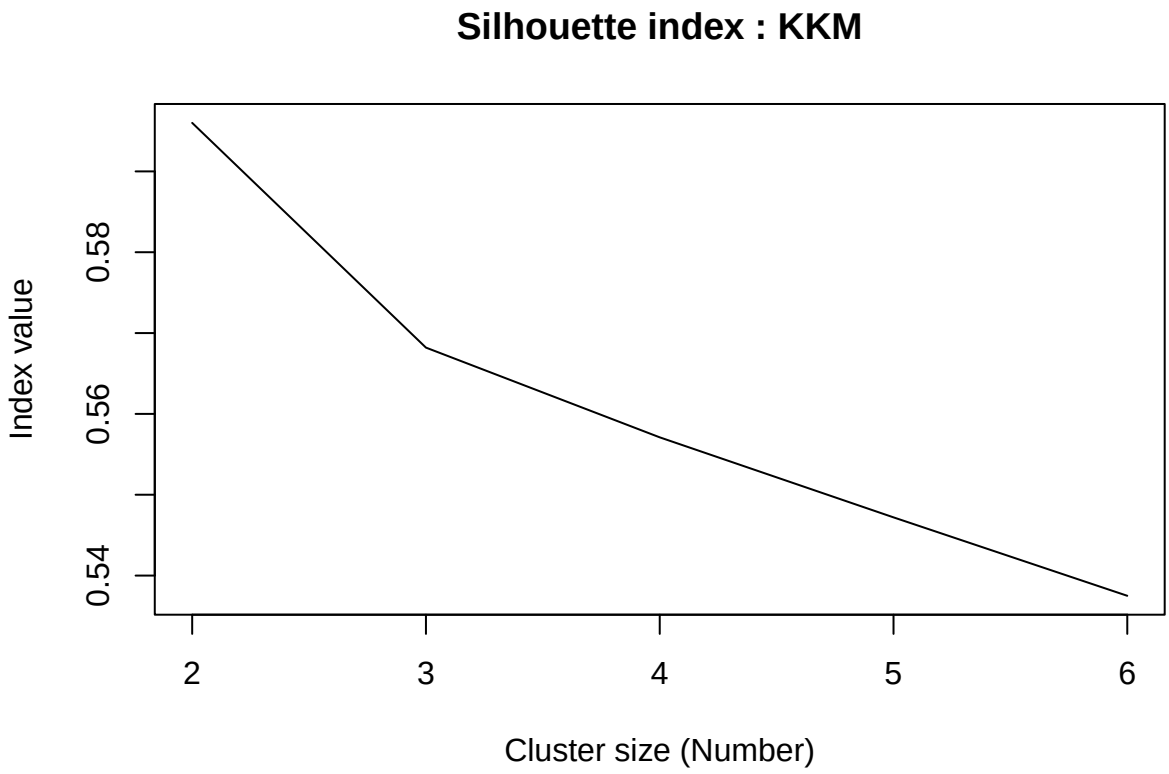


The graphs indicate that 4 is the optimal number of k. Beyond $k = 4$, increasing the number of k does not result in a significant reduction in the sum of squares within (strong internal cohesion) nor does it result in a significant increase in the sum of squares between and the ratio of between to the total sum of squares (maximum external separation).

We will calculate the optimal number of clusters using the Silhouette index by the function `NbClust`. Then we will plot the silhouette index as a function of the cluster size:

```
cluster_sizes_km_s <- NbClust(data = nba,
                              #  it will probably be more difficult to interpret clusters beyond
                              #this number
                              max.nc = 6,
```

```
                          method = 'kmeans',
                          index = 'silhouette')

plot(names(cluster_sizes_km_s$All.index),
     cluster_sizes_km_s$All.index,
     main = 'Silhouette index : KKM',
     type = 'l',
     xlab = "Cluster size (Number)",
     ylab = "Index value")
```

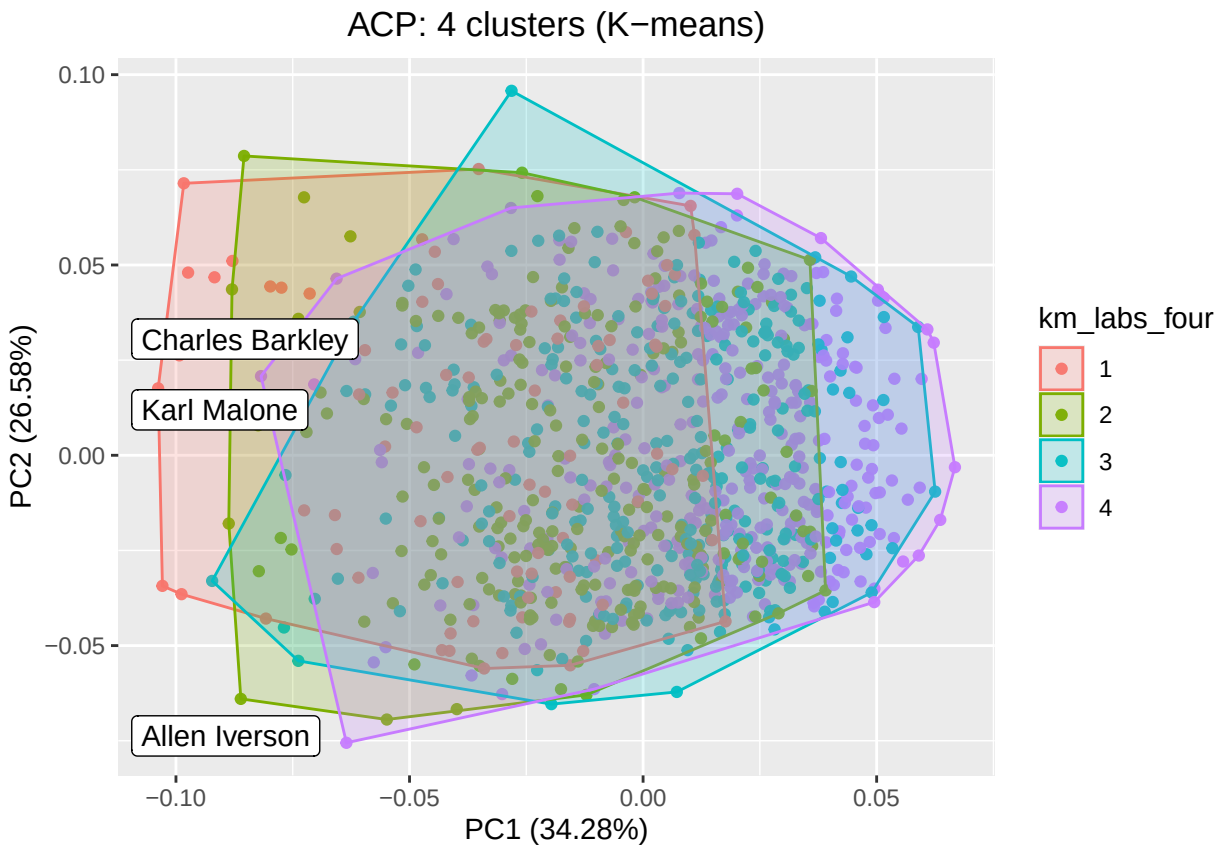


The Silhouette index plot also shows that 4 is the optimal number of k. The characteristics of each cluster can also be visualized by combining clustering and PCA in the following graph.

```
# Compute the k-means for k = 4 clusters :
km_k <- 4
km_four <- kmeans(x = nba,
            centers = km_k,
            nstart = 100,
            algorithm = 'Hartigan-Wong')

# Add cluster assignments to the original dataset:
historical_players.df$km_labs_four <- factor(km_four$cluster)

# Swap labels 3 and 4 so that label 4 is the best player:
#nba$km_labs_four <- factor(ifelse(nba$km_labs_four == 4, 3,
#                          ifelse(nba$km_labs_four == 3, 4,
 #                         nba$km_labs_four)))

# Plotting K-Means clusters in PC space:
```

```
# Add player labels to the chart - players who have played more than 36 min
#per game or less than 3 min per game:
km_labels <- ifelse(historical_players.df$min >= 36 | historical_players.df$min <= 3,
                        as.character(historical_players.df$player_name), '' )
# clustering visualization
plot_pca(km_five, data = historical_players.df, frame = TRUE, colour = 'km_labs_four',
          title = paste0('ACP: ', km_k, ' clusters (K-means)'),
          label = km_labels
         #leg_title = 'Clusters'
         )
```

```
## Warning: ggrepel: 26 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



This profiling graph shows the following findings on players playing more than 36 minutes or less than 3 minutes per game:

- **Cluster 1**: is dominated by Kareem Abdul-Jabbar and features mostly players who have played less than 36 minutes or more than 3 minutes per game.
- **Cluster 2**: is dominated by Charles Barkley and features fewer players who have played less than 36 minutes or more than 3 minutes per game.
- **Cluster 3**: is dominated by Karl Malone and has about the same number of players who played less than 36 minutes or more than 3 minutes per game.
- **Cluster 4**: is dominated by Allen Iverson and has mostly players who have played more than 36 minutes or less than 3 minutes per game.

```
km_labs <- data.frame(table(historical_players.df$km_labs_four))
colnames(km_labs) <- col_names
```

```
#Affichage le tableau
km_labs %>%
  kable("latex", booktabs = T,
        caption = "The number of players in each cluster according to the k-means method")%>%
  kable_styling(latex_options = c("striped","HOLD_position"),font_size = 20)
```

Table 8: The number of players in each cluster according to the k-means method

Clusters	Count
1	88
2	249
3	308
4	357

```
nba_km_avg <- historical_players.df %>%
  select(km_labs_four, min, pts, oreb, dreb,
         ast, blk, stl) %>%
  group_by(km_labs_four) %>%
  summarise_all(list(mean))

nba_km_avg %>%
  kable("latex", booktabs = T,
        caption = "Average stats of NBA players in the clusters of the k-mean method")%>%
  kable_styling(latex_options = c("striped", "scale_down","HOLD_position"))
```

Table 9: Average stats of NBA players in the clusters of the k-mean method

km_labs_four	min	pts	oreb	dreb	ast	blk	stl
1	30.65066	14.514584	1.857508	4.474152	3.104956	0.7915108	1.0289972
2	28.22468	12.584718	1.411883	3.501997	2.925397	0.6033321	0.9736163
3	24.60549	10.187075	1.253143	2.937565	2.412079	0.4683629	0.8453572
4	22.15491	8.887519	1.197935	2.763746	1.953454	0.4582003	0.7340438

The following are the means of the original variables (`min`, `pts`, `oreb`, `dreb`, `ast`, `blk`, `stl`) for each player grouping created. We have reduced the data to fewer dimensions to get a clearer grouping, but we still retain all the original information to understand the player groupings in the context of understandable basketball statistics.

Cross-tabulation of label evaluations to compare hierarchical and K-means methods:

```
xtab_hcl_km <- xtabs(~historical_players.df$hcl_ward_four + historical_players.df$km_labs_four)

xtab_hcl_km %>%
  kable("latex", booktabs = T,
        caption = "Cross-tabulation between the hierarchical and K-means methods")%>%
```

```
kable_styling(latex_options = c("striped","HOLD_position"),font_size = 20)
```

Table 10: Cross-tabulation between the hierarchical and K-means methods

1	2	3	4
88	102	0	0
0	0	270	35
0	147	38	0
0	0	0	322

According to the table above, the K-Means method with four clusters seems to be the optimal solution, as it produced a clearer separation of players based on their overall skills.

Results, discussion and conclusion

In conclusion, we showed an unsupervised learning analysis of NBA player statistics. We used K-means and hierarchical clustering and PCA dimensionality reduction. We can summarize that:

- Dimensionality reduction can be applied to this data by choosing a number of PCs from 24 PCs. In this case, we chose 2 PCs to reduce the dimensionality by 39.14% while keeping 60.86% of the information.
- Hierarchical clustering and K-means are used on this dataset to rank players based on their playing time per game. The clusters help distinguish between players who play more than 36 minutes or less than 3 minutes per game, such as Allen Iverson, Karl Malone, Charles Barkley and Abdul-Jabbar.

References

[1] Factor analysis of mixed data in R

[2] PCA For Dimensionality Reduction

[3] Predicting breast cancer using PCA + LDA in R

[4] K-Means Clustering and PCA with Visualization

[5] Dimensionality Reduction on Cereals

[6] Clustering NBA Player Types: A Tutorial on K-Means, Gaussian Mixture Models, Principal Component Analysis, and Graphical Networks

[7] Clustering Lab – NBA Salaries Part

[8] Clustering Basketball Players by Positio

[9] Redefining NBA Player Classifications using Clustering

[10] NBA Clustering Analysis

[11] PCA and Clustering Analysis of NBA Shot Selections

[12] NBA data cleaning (my first kernel)

[13] NBA data preparation

[14] PCA on NBA Players with R

[15] An Introduction to corrplot Package