R Notebook

Code

```
library(tidyverse)
```

```
[30m-- [1mAttaching packages[22m --------------------------------------- tidyve
rse 1.2.1 --[39m
[30m[32mv[30m [34mggplot2[30m 3.1.0     [32mv[30m [34mpurrr  [30m 0.2.5
[32mv[30m [34mtibble [30m 1.4.2     [32mv[30m [34mdplyr  [30m 0.7.8
[32mv[30m [34mtidyr  [30m 0.8.1     [32mv[30m [34mstringr[30m 1.3.1
[32mv[30m [34mreadr  [30m 1.1.1     [32mv[30m [34mforcats[30m 0.3.0[39m
[30m-- [1mConflicts[22m ------------------------------------------ tidyverse_co
nflicts() --
[31mx[30m [34mdplyr[30m::[32mfilter()[30m masks [34mstats[30m::filter()
[31mx[30m [34mdplyr[30m::[32mlag()[30m    masks [34mstats[30m::lag()[39m
```

```
library("FactoMineR")
```

```
package <U+393C><U+3E31>FactoMineR<U+393C><U+3E32> was built under R version 3.
5.3
```

```
library("factoextra")
```

```
package <U+393C><U+3E31>factoextra<U+393C><U+3E32> was built under R version 3.
5.3Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at http
s://goo.gl/13EFCZ
```

```
library(ggcorrplot)
```

```
package <U+393C><U+3E31>ggcorrplot<U+393C><U+3E32> was built under R version 3.
5.3
```

```
library(missMDA)
```

```
package <U+393C><U+3E31>missMDA<U+393C><U+3E32> was built under R version 3.5.3
```

```
setwd("C:/Users/Veeru/Desktop/ML/PCA")
getwd()
```

```
[1] "C:/Users/Veeru/Desktop/ML/PCA"
```

```
fifa <- read.csv("FIFA.csv", sep = ",")
str(fifa)
```

```
'data.frame':   17981 obs. of  76 variables:
 $ Name                     : Factor w/ 16975 levels "A. ÃmÃ¼r","A. Ãrn Arnarso
n",..: 3203 9648 12428 9850 11176 13779 4211 4519 15725 5805 ...
 $ Age                      : int  32 30 25 30 31 28 26 26 27 29 ...
 $ Nationality              : Factor w/ 165 levels "Afghanistan",..: 122 6 19 1
59 59 121 139 13 59 6 ...
 $ Overall                  : int  94 93 92 92 92 91 90 90 90 90 ...
 $ Potential                : int  94 93 94 92 92 91 92 91 90 90 ...
 $ Club                     : Factor w/ 647 levels "1. FC Heidenheim",..: 470 2
24 435 224 227 227 382 146 470 337 ...
 $ Value_1                  : num  9.55e+07 1.05e+08 1.23e+08 9.70e+07 6.10e+0
7 9.20e+07 6.45e+07 9.05e+07 7.90e+07 7.70e+07 ...
 $ Value_2                  : Factor w/ 3 levels "0","K","M": 3 3 3 3 3 3 3 3
3 3 ...
 $ Wage_1                   : num  565000 565000 280000 510000 230000 355000 21
5000 295000 340000 275000 ...
 $ Wage_2                   : Factor w/ 2 levels "0","K": 2 2 2 2 2 2 2 2 2
2 ...
 $ Special                  : int  2228 2154 2100 2291 1493 2143 1458 2096 216
5 1961 ...
 $ Acceleration             : int  89 92 94 88 58 79 57 93 60 78 ...
 $ Aggression               : int  63 48 56 78 29 80 38 54 60 50 ...
 $ Agility                  : int  89 90 96 86 52 78 60 93 71 75 ...
 $ Balance                  : int  63 95 82 60 35 80 43 91 69 69 ...
 $ Ball.control             : int  93 95 95 91 48 89 42 92 89 85 ...
 $ Composure                : int  95 96 92 83 70 87 64 87 85 86 ...
 $ Crossing                 : int  85 77 75 77 15 62 17 80 85 68 ...
 $ Curve                    : int  81 89 81 86 14 77 21 82 85 74 ...
 $ Dribbling                : int  91 97 96 86 30 85 18 93 79 84 ...
 $ Finishing                : int  94 95 89 94 13 91 13 83 76 91 ...
 $ Free.kick.accuracy       : int  76 90 84 84 11 84 19 79 84 62 ...
 $ GK.diving                : int  7 6 9 27 91 15 90 11 10 5 ...
 $ GK.handling              : int  11 11 9 25 90 6 85 12 11 12 ...
 $ GK.kicking               : int  15 15 15 31 95 12 87 6 13 7 ...
 $ GK.positioning           : int  14 14 15 33 91 8 86 8 7 5 ...
 $ GK.reflexes              : int  11 8 11 37 89 10 90 8 10 10 ...
 $ Heading.accuracy         : int  88 71 62 77 25 85 21 57 54 86 ...
 $ Interceptions            : int  29 22 36 41 30 39 30 41 85 20 ...
 $ Jumping                  : int  95 68 61 69 78 84 67 59 32 79 ...
 $ Long.passing             : int  77 87 75 64 59 65 51 81 93 59 ...
 $ Long.shots               : int  92 88 77 86 16 83 12 82 90 82 ...
 $ Marking                  : int  22 13 21 30 10 25 13 25 63 12 ...
 $ Penalties                : int  85 74 81 85 47 81 40 86 73 70 ...
 $ Positioning              : int  95 93 90 92 12 91 12 85 79 92 ...
 $ Reactions                : int  96 95 88 93 85 91 88 85 86 88 ...
 $ Short.passing            : int  83 88 81 83 55 83 50 86 90 75 ...
 $ Shot.power               : int  94 85 80 87 25 88 31 79 87 88 ...
 $ Sliding.tackle           : int  23 26 33 38 11 19 13 22 69 18 ...
```

```
 $ Sprint.speed             : int  91 87 90 77 61 83 58 87 52 80 ...
 $ Stamina                  : int  92 73 78 89 44 79 40 79 77 72 ...
 $ Standing.tackle          : int  31 28 24 45 10 42 21 27 82 22 ...
 $ Strength                 : int  80 59 53 80 83 84 64 65 74 85 ...
 $ Vision                   : int  85 90 80 84 70 78 68 86 88 70 ...
 $ Volleys                  : int  88 85 83 88 11 87 13 79 82 88 ...
 $ CAM                      : int  89 92 88 87 NA 84 NA 88 83 81 ...
 $ CB                       : int  53 45 46 58 NA 57 NA 47 72 46 ...
 $ CDM                      : int  62 59 59 65 NA 62 NA 61 82 52 ...
 $ CF                       : int  91 92 88 88 NA 87 NA 87 81 84 ...
 $ CM                       : int  82 84 79 80 NA 78 NA 81 87 71 ...
 $ ID                       : int  20801 158023 190871 176580 167495 188545 193
080 183277 182521 167664 ...
 $ LAM                      : int  89 92 88 87 NA 84 NA 88 83 81 ...
 $ LB                       : int  61 57 59 64 NA 58 NA 59 76 51 ...
 $ LCB                      : int  53 45 46 58 NA 57 NA 47 72 46 ...
 $ LCM                      : int  82 84 79 80 NA 78 NA 81 87 71 ...
 $ LDM                      : int  62 59 59 65 NA 62 NA 61 82 52 ...
 $ LF                       : int  91 92 88 88 NA 87 NA 87 81 84 ...
 $ LM                       : int  89 90 87 85 NA 82 NA 87 81 79 ...
 $ LS                       : int  92 88 84 88 NA 88 NA 82 77 87 ...
 $ LW                       : int  91 91 89 87 NA 84 NA 88 80 82 ...
 $ LWB                      : int  66 62 64 68 NA 61 NA 64 78 55 ...
 $ Preferred.Positions      : Factor w/ 15 levels "CAM","CB","CDM",..: 15 13 9
15 6 15 6 9 3 15 ...
 $ RAM                      : int  89 92 88 87 NA 84 NA 88 83 81 ...
 $ RB                       : int  61 57 59 64 NA 58 NA 59 76 51 ...
 $ RCB                      : int  53 45 46 58 NA 57 NA 47 72 46 ...
 $ RCM                      : int  82 84 79 80 NA 78 NA 81 87 71 ...
 $ RDM                      : int  62 59 59 65 NA 62 NA 61 82 52 ...
 $ RF                       : int  91 92 88 88 NA 87 NA 87 81 84 ...
 $ RM                       : int  89 90 87 85 NA 82 NA 87 81 79 ...
 $ RS                       : int  92 88 84 88 NA 88 NA 82 77 87 ...
 $ RW                       : int  91 91 89 87 NA 84 NA 88 80 82 ...
 $ RWB                      : int  66 62 64 68 NA 61 NA 64 78 55 ...
 $ ST                       : int  92 88 84 88 NA 88 NA 82 77 87 ...
 $ Preferred.Positions_match: logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Position                 : Factor w/ 4 levels "DEF","FWD","GK",..: 2 4 4 2
3 2 3 4 4 2 ...
 $ Wage_Log10               : num  5.75 5.75 5.45 5.71 5.36 ...
```

The fifa data set above has 17981 obs. of 76 variables. That means we have data about 17981 players in all 76 variables which are players skills related.

```
colna <- data.frame(colSums(is.na(fifa)))
colnames(colna) <- c("number_of_na")
colna
```

The fifa csv file as many NaN's. The column wise count of NaN's is stored in the dataframe"colna" below. There are 76 columns in "fifa"" dataframe which have NaN's.

```
names(fifa)[sapply(fifa, anyNA)]
```

```
 [1] "Club"           "Value_1"        "Wage_1"         "GK.positioning"
 [5] "CAM"            "CB"             "CDM"            "CF"
 [9] "CM"             "LAM"            "LB"             "LCB"
[13] "LCM"            "LDM"            "LF"             "LM"
[17] "LS"             "LW"             "LWB"            "RAM"
[21] "RB"             "RCB"            "RCM"            "RDM"
[25] "RF"             "RM"             "RS"             "RW"
[29] "RWB"            "ST"             "Wage_Log10"
```

All the above variables have Na's. As we know

Principal component analysis can be done on numerical continuous variables only.

Thus club variable which is a factor having Na will not be included. but rest all variables are continuous numeric variables, which we may need for PCA.

Since i have limited knowledge about football framing the question without having the understanding about the subject is difficult for me, so I will be following this web link as my reference.

https://blog.exploratory.io/using-pca-to-see-which-countries-have-better-players-for-world-cup-games-a72f91698b95 (https://blog.exploratory.io/using-pca-to-see-which-countries-have-better-players-for-world-cup-games-a72f91698b95)

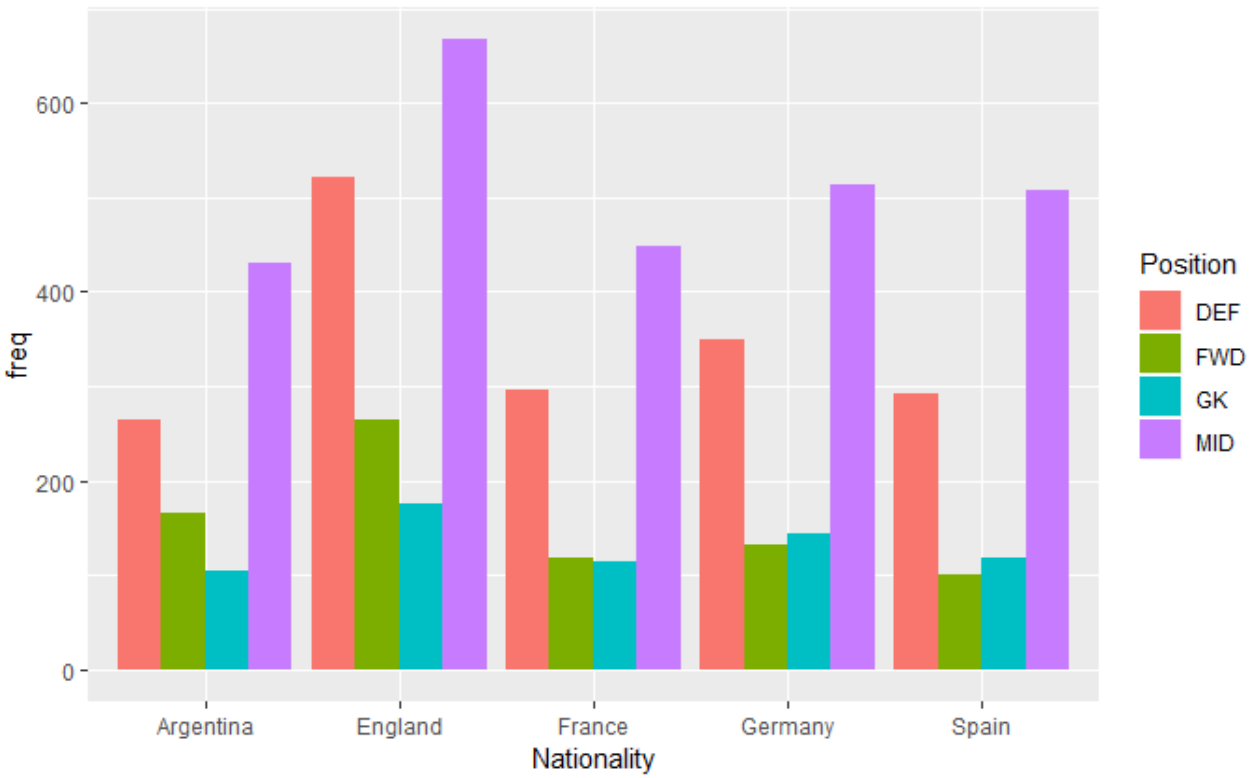
```
library(plyr)
```

```
------------------------------------------------------------------------------
--
You have loaded plyr after dplyr - this is likely to cause problems.
If you need functions from both plyr and dplyr, please load plyr first, then dp
lyr:
library(plyr); library(dplyr)
------------------------------------------------------------------------------
--

Attaching package: <U+393C><U+3E31>plyr<U+393C><U+3E32>

The following objects are masked from <U+393C><U+3E31>package:dplyr<U+393C><U+3
E32>:

    arrange, count, desc, failwith, id, mutate, rename, summarise,
    summarize

The following object is masked from <U+393C><U+3E31>package:purrr<U+393C><U+3E3
2>:

    compact
```

```
count(fifa, c("Nationality")) %>% arrange(desc(freq))
```

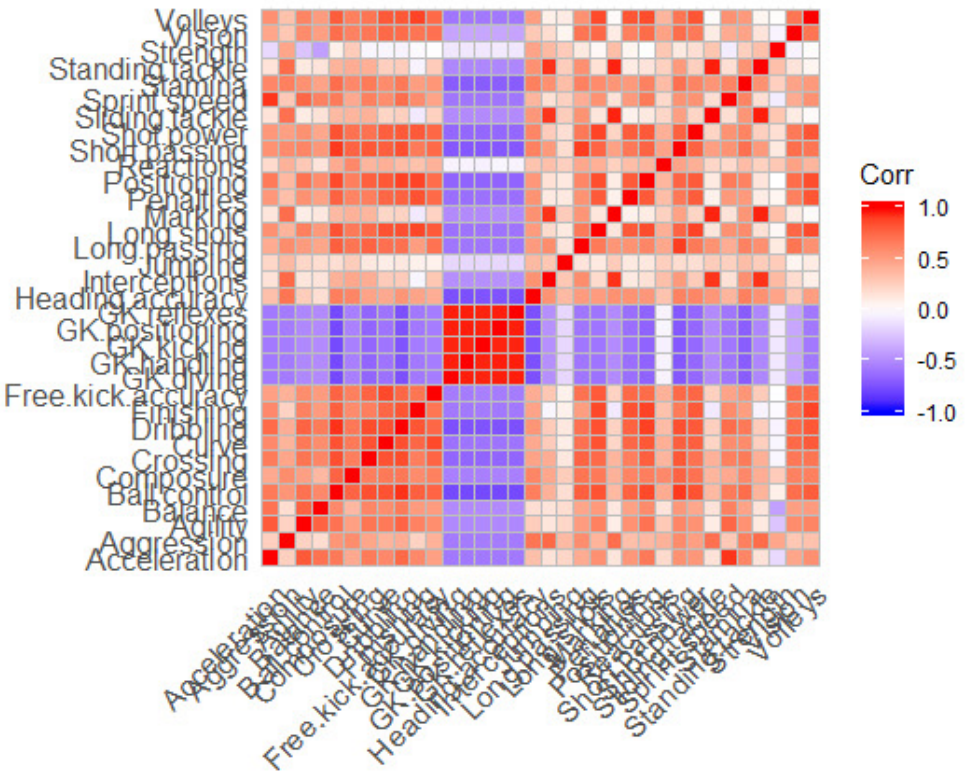
England has the higest number of players following germany, spain,france and argentina coming in top 5

```
England <- filter(fifa, Nationality %in% c("England")) %>% select(Nationality,
Position) %>% count("Position") %>% mutate(Nationality="England")
Germany <- filter(fifa, Nationality %in% c("Germany")) %>% select(Nationality,
Position) %>% count("Position")%>% mutate(Nationality="Germany")
Spain <- filter(fifa, Nationality %in% c("Spain")) %>% select(Nationality, Posi
tion) %>% count("Position")%>% mutate(Nationality="Spain")
France <- filter(fifa, Nationality %in% c("France")) %>% select(Nationality, Po
sition) %>% count("Position")%>% mutate(Nationality="France")
Argentina <- filter(fifa, Nationality %in% c("Argentina")) %>% select(Nationali
ty, Position) %>% count("Position")%>% mutate(Nationality="Argentina")
df1 <- rbind(England, Germany, Spain, France, Argentina)
plot <-ggplot(df1, aes(Nationality, freq))
plot +geom_bar(stat = "identity", aes(fill = Position), position = "dodge")
```



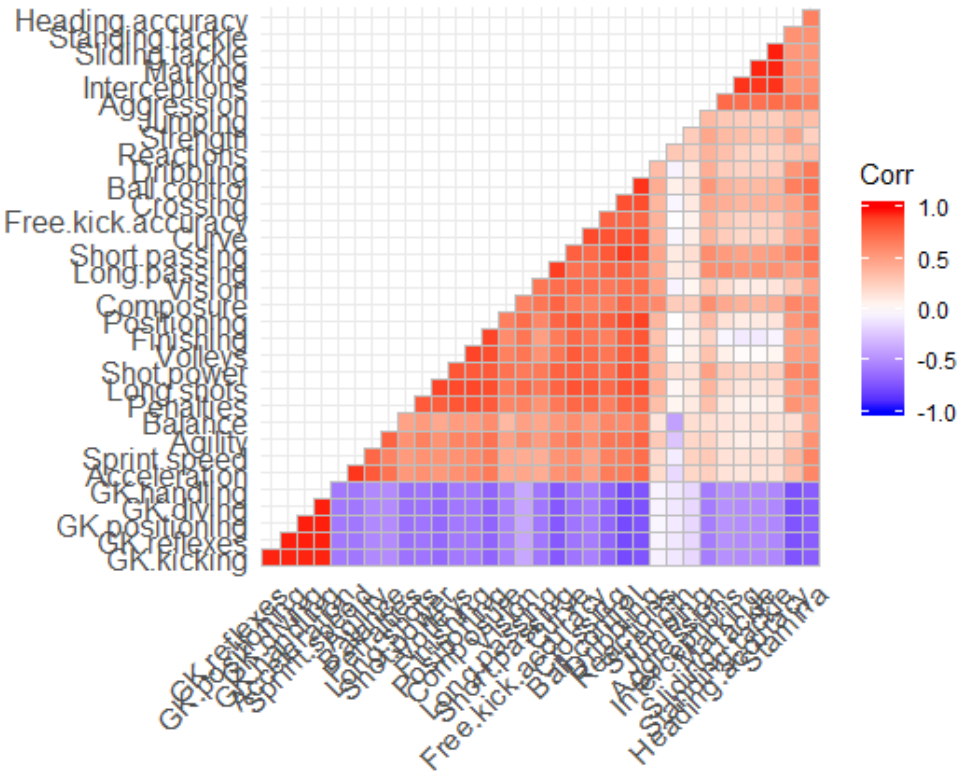
The plot above shows the distribution of players by positions. To understand about relationship with skills of players a correlation matrix is a good idea.

```
correl <- cor(fifa[,12:45], use = "complete.obs")
ggcorrplot(correl)
```



creating a correlation matrix of the variables 12 to 45, which have the skills of each player.As I have found this dataset has Na's, i am going to use only complete observation for the correlation graph.The plot when opened in a seperate window is big and we can clearly see that the GK(goal keeping) skills of players are negatively correlated with all the other skills, that means players with high GK skills have lower acceleration and so on. We can also observe that strength skill is negatively correlated to Acceleration, agility and balance.

```
ggcorrplot_clustered <- ggcorrplot(correl, hc.order = TRUE, type = "lower")
ggcorrplot_clustered
```



This is hierarchical clustering on the correlation matrix and this produces same information as the plot before. This plot is much easier to view and understand.

The fifa dataset has Na's in the GK.positioning variable, we cannot skip the missing values since it is a risky option that leads to unreliable PCA model. To impute the missing values, using missMDA package.

```
fifa_ncp <- estim_ncpPCA(fifa[,12:45])
```

'estim_ncpPCA()' is necessary for determining the optimum number of PC's, which is 5 as shown below.

```
fifa_ncp$ncp
```

```
[1] 5
```

```
complete_fifa <- imputePCA(fifa[,12:45], ncp = fifa_ncp$ncp, scale = TRUE)
```

'imputePCA()' outputs the dataset without Na's, here ncp is the number of principal components

scale if TRUE, the data are scaled to unit variance before the analysis. This standardization to the same scale avoids some variables to become dominant just because of their large measurement units. It makes variables comparable.

```
fifanew <- complete_fifa$completeObs
head(fifanew)
```

```
     Acceleration Aggression Agility Balance Ball.control Composure Crossing Cu
rve
[1,]           89         63      89      63           93        95       8
5    81
[2,]           92         48      90      95           95        96       7
7    89
[3,]           94         56      96      82           95        92       7
5    81
[4,]           88         78      86      60           91        83       7
7    86
[5,]           58         29      52      35           48        70       1
5    14
[6,]           79         80      78      80           89        87       6
2    77
     Dribbling Finishing Free.kick.accuracy GK.diving GK.handling GK.kicking
[1,]        91        94                 76         7          11         15
[2,]        97        95                 90         6          11         15
[3,]        96        89                 84         9           9         15
[4,]        86        94                 84        27          25         31
[5,]        30        13                 11        91          90         95
[6,]        85        91                 84        15           6         12
     GK.positioning GK.reflexes Heading.accuracy Interceptions Jumping
[1,]             14          11                88            29      95
[2,]             14           8                71            22      68
[3,]             15          11                62            36      61
[4,]             33          37                77            41      69
[5,]             91          89                25            30      78
[6,]              8          10                85            39      84
     Long.passing Long.shots Marking Penalties Positioning Reactions Short.pass
ing
[1,]           77         92      22        85          95        9
6            83
[2,]           87         88      13        74          93        9
5            88
[3,]           75         77      21        81          90        8
8            81
[4,]           64         86      30        85          92        9
3            83
[5,]           59         16      10        47          12        8
5            55
[6,]           65         83      25        81          91        9
1            83
     Shot.power Sliding.tackle Sprint.speed Stamina Standing.tackle Strength
[1,]         94             23           91      92              31       80
[2,]         85             26           87      73              28       59
[3,]         80             33           90      78              24       53
[4,]         87             38           77      89              45       80
[5,]         25             11           61      44              10       83
```

```
[6,]         88             19           83      79               42       84
     Vision Volleys
[1,]     85      88
[2,]     90      85
[3,]     80      83
[4,]     84      88
[5,]     70      11
[6,]     78      87
```

fifanew is the dataset which has imputed Na values in GK.positioning variable. The resulting “fifanew” will be used as an argument for PCA().

```
fifaold <- fifa %>% select(Nationality,Position)
fifa1 <- cbind(fifaold,fifanew)
fifa1
```

I am combining some of the categorical variables from original “fifa” data set with “fifa1”.I am unaware of future usage of these categorical variables in principalcomponent analysis, but for now i will be keeping them. Idea is to use them to plot ellipsoids

```
pca_output <- PCA(fifa1, quali.sup=1:2, graph = FALSE)
```

By default, the function PCA() [in FactoMineR], standardizes the data automatically during the PCA, when scale =TRUE ; so we don’t need do this transformation before the PCA.

However when imputing we have already scaled the data to be standardized

By default, PCA() generates 2 graphs and extracts the first 5 PCs.we can for instance use ncp=3 argument in PCA() to manually set the number of dimensions to 3.

To ignore some of the original variables or individuals in building a PCA model by supplying PCA() with the ind.sup argument for supplementary individuals and quanti.sup or quali.sup for quantitative and qualitative variables respectively. Supplementary individuals and variables are rows and variables of the original data ignored while building the model.

since “fifa1” have 2 variables which are categorical i have used code to supress it, i have also supressed the plots and will be plotting same later.

```
pca_output
```

```
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 17981 individuals, described by 36 variables
*The results are available in the following objects:

   name                 description
1  "$eig"               "eigenvalues"
2  "$var"               "results for the variables"
3  "$var$coord"         "coord. for the variables"
4  "$var$cor"           "correlations variables - dimensions"
5  "$var$cos2"          "cos2 for the variables"
6  "$var$contrib"       "contributions of the variables"
7  "$ind"               "results for the individuals"
8  "$ind$coord"         "coord. for the individuals"
9  "$ind$cos2"          "cos2 for the individuals"
10 "$ind$contrib"       "contributions of the individuals"
11 "$quali.sup"         "results for the supplementary categorical variables"
12 "$quali.sup$coord"   "coord. for the supplementary categories"
13 "$quali.sup$v.test" "v-test of the supplementary categories"
14 "$call"              "summary statistics"
15 "$call$centre"       "mean of the variables"
16 "$call$ecart.type"   "standard error of the variables"
17 "$call$row.w"        "weights for the individuals"
18 "$call$col.w"        "weights for the variables"
```

Above are all the components under PCA

```
get_eigenvalue(pca_output)
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	18.80561898	55.31064405	55.31064
Dim.2	5.04846963	14.84844009	70.15908
Dim.3	2.35595595	6.92928221	77.08837
Dim.4	1.70576176	5.01694635	82.10531
Dim.5	1.33726023	3.93311832	86.03843
Dim.6	0.62370022	1.83441240	87.87284
Dim.7	0.43096454	1.26754275	89.14039
Dim.8	0.37127716	1.09199163	90.23238
Dim.9	0.30899112	0.90879741	91.14118
Dim.10	0.27747757	0.81611050	91.95729
Dim.11	0.25383622	0.74657712	92.70386
Dim.12	0.23603702	0.69422654	93.39809
Dim.13	0.22651233	0.66621274	94.06430
Dim.14	0.21607826	0.63552429	94.69983
Dim.15	0.20550402	0.60442360	95.30425
Dim.16	0.18801792	0.55299389	95.85724
Dim.17	0.17641659	0.51887232	96.37612
Dim.18	0.14472242	0.42565417	96.80177
Dim.19	0.13318562	0.39172240	97.19349
Dim.20	0.12830872	0.37737860	97.57087
Dim.21	0.11233565	0.33039896	97.90127
Dim.22	0.09671471	0.28445502	98.18573
Dim.23	0.08926325	0.26253897	98.44826
Dim.24	0.07902821	0.23243591	98.68070
Dim.25	0.07601880	0.22358470	98.90428
Dim.26	0.07143769	0.21011085	99.11440
Dim.27	0.06399977	0.18823462	99.30263
Dim.28	0.04377478	0.12874934	99.43138
Dim.29	0.03897918	0.11464465	99.54602
Dim.30	0.03892418	0.11448288	99.66051
Dim.31	0.03230003	0.09500009	99.75551
Dim.32	0.03124147	0.09188668	99.84739
Dim.33	0.02720388	0.08001142	99.92741
Dim.34	0.02468214	0.07259454	100.00000

the eigenvalues measure the amount of variation retained by each principal component. Eigenvalues are large for the first PCs and small for the subsequent PCs.

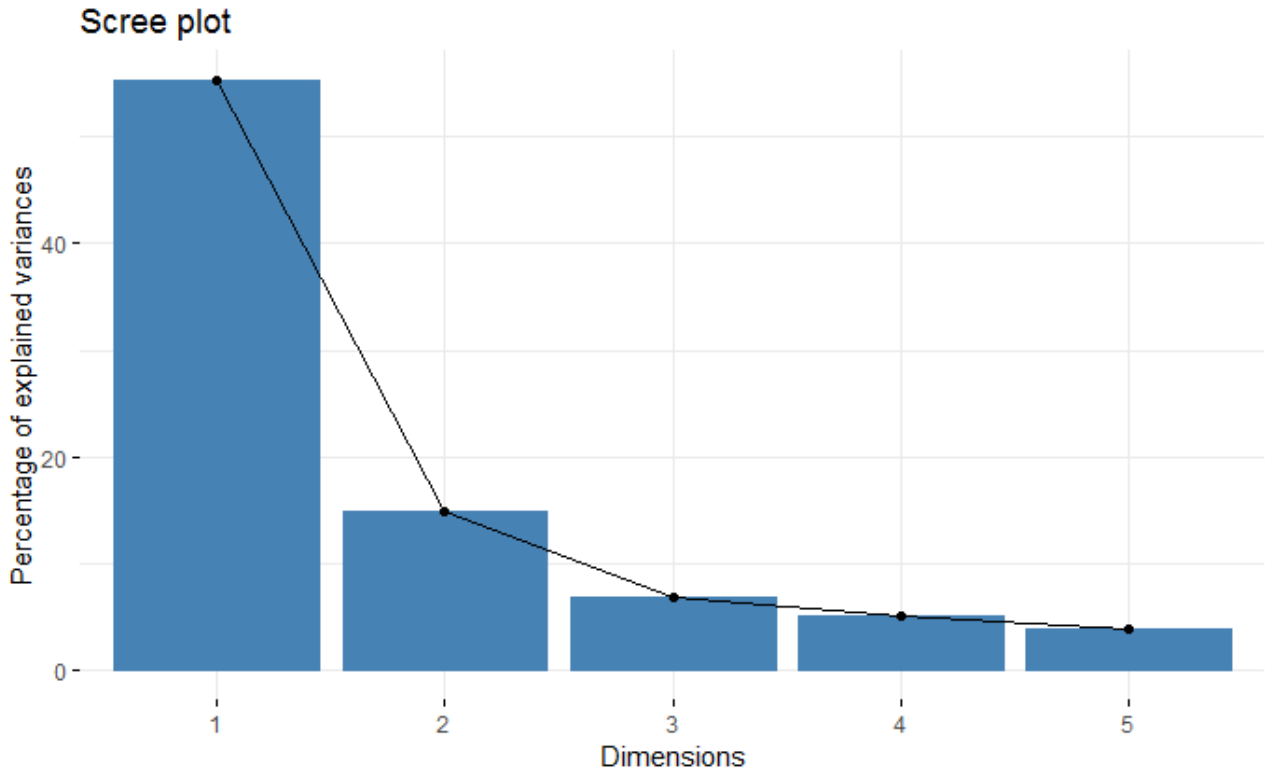
The sum of all the eigenvalues give a total variance of 34.

The proportion of variation explained by each eigenvalue is given in the second column. For example, 18.80 divided by 34 equals 0.5531, or, about 55.31% of the variation is explained by this first eigenvalue.

The cumulative percentage explained is obtained by adding the successive proportions of variation explained to obtain the running total. For instance, 55.31% plus 14.84% equals 70.15%, and so forth.

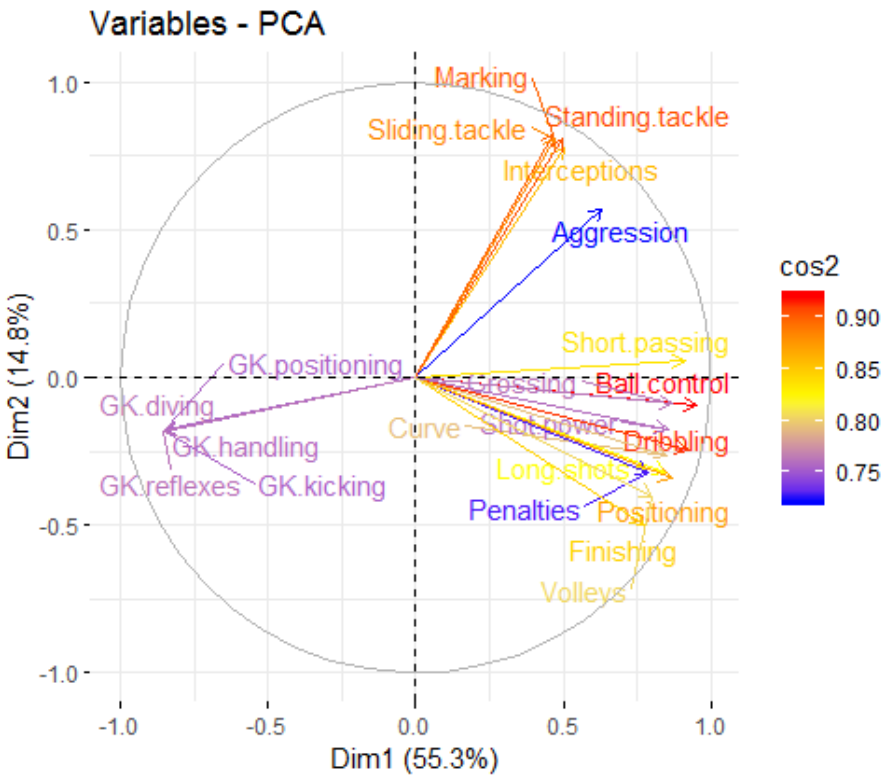
An eigenvalue > 1 indicates that PCs account for more variance than others

```
fviz_screeplot(pca_output, ncp = 5)
```



the screeplot above confirms that PC1/Dim1 contributes approximately 55% of variance and pc2 contibutes 14%

```
fviz_pca_var(pca_output,col.var="cos2", select.var = list(cos2 = 0.7), gradien
t.cols = c("blue", "yellow", "red"), repel = TRUE)
```



Create a factor map using fviz_pca_var() for the variables with cos2 higher than 0.7.

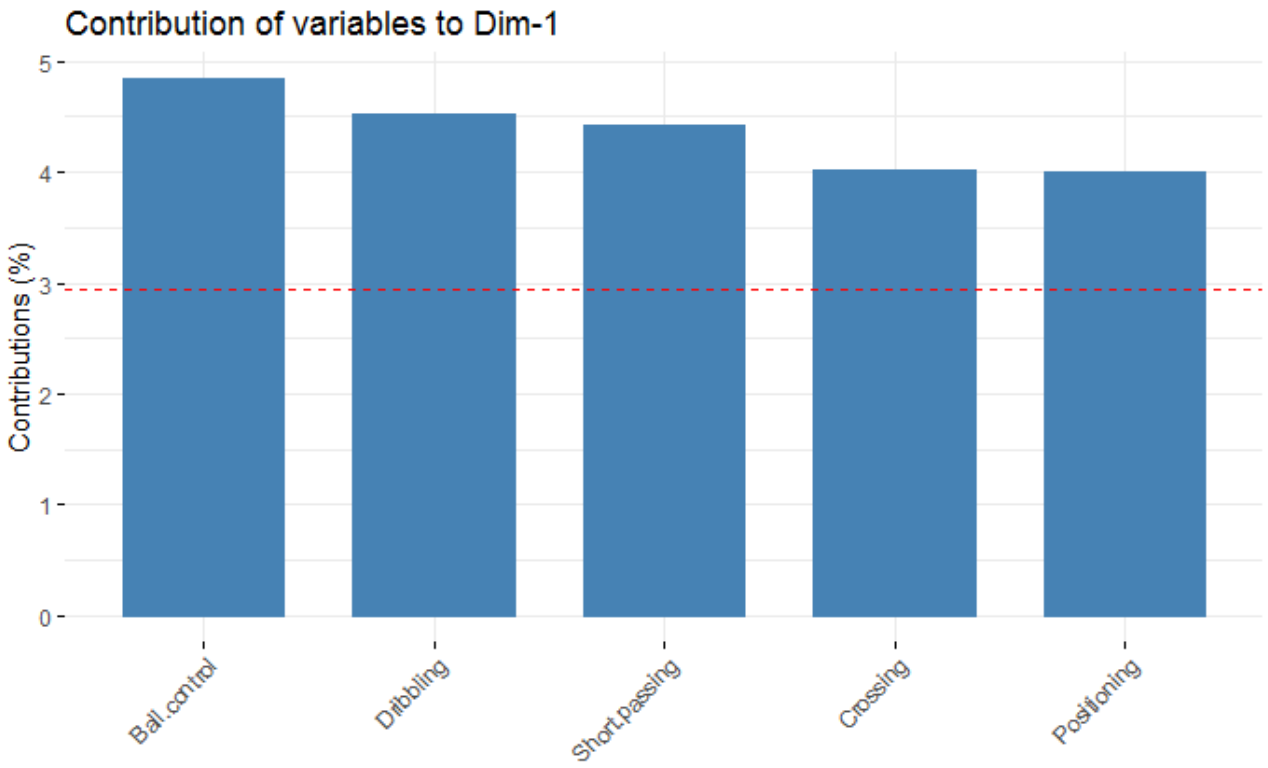
cos2 shows how accurate the representation of your variables or individuals on the PC plane is.

To create the individuals/rows factor map using fviz_pca_ind(), code:“fviz_pca_ind(pca_output, select.ind = list(cos2 = 0.7), repel = TRUE)”

The distance between variables and the origin measures the quality of the variables on the factor map. Variables that are away from the origin are well represented on the factor map

Positively correlated variables are grouped together away from negative correlated ones, in this case it is GK vs rest all.

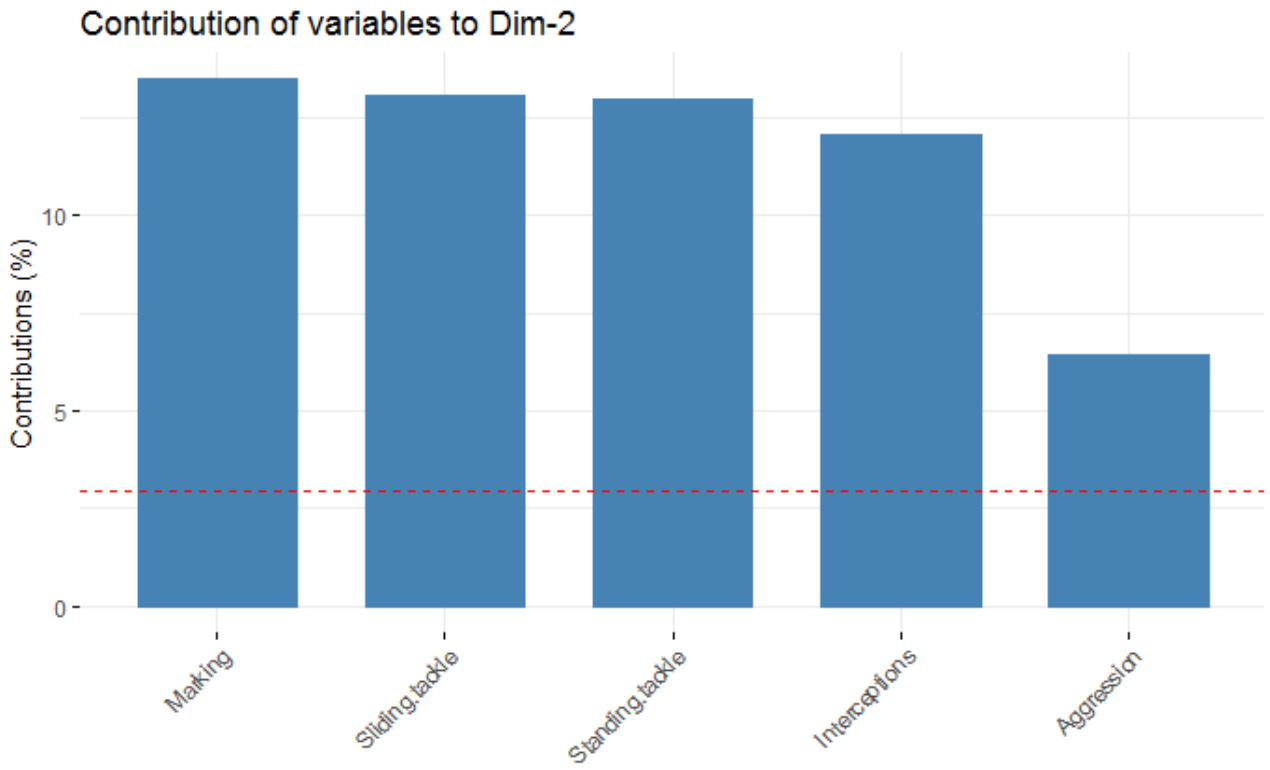
```
fviz_contrib(pca_output, choice = "var", axes = 1, top = 5)
```



barplot for the variables with the highest contributions to the 1st PC

The red dashed line on the bar graph indicates the expected average contribution

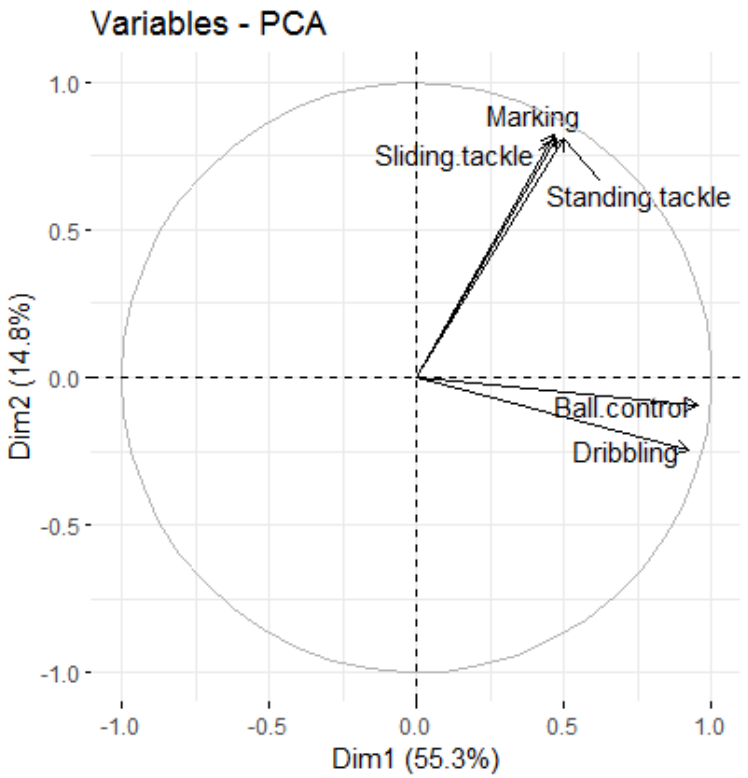
```
fviz_contrib(pca_output, choice = "var", axes = 2, top = 5)
```



barplot for the variables with the highest contributions to the 2nd PC.

the dotted line here much below suggesting the contibrution of these variables is more than average.

```
fviz_pca_var(pca_output, select.var = list(contrib = 5), repel = TRUE)
```

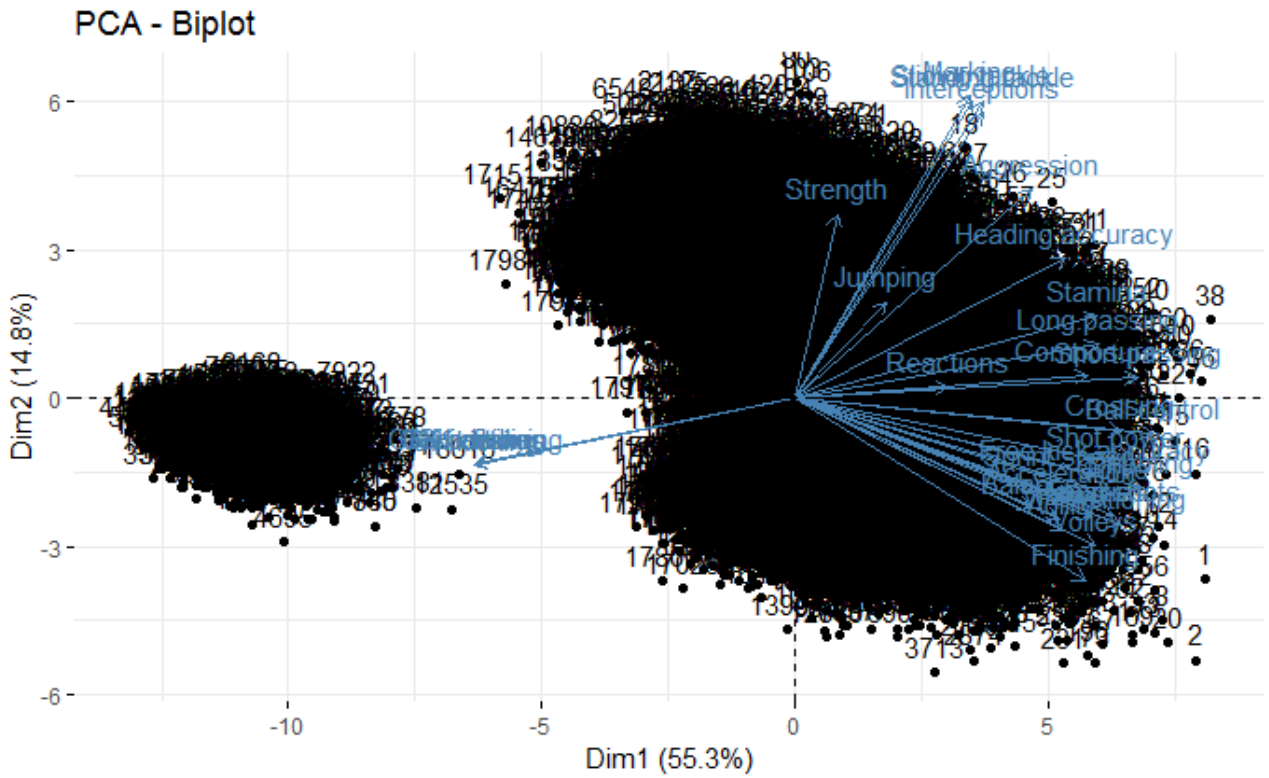


contributions of the variables essentially signify their importance for the construction of a given principal component.

factor map for the top 5 variables with the highest contributions.

Ball.control and Dribbling are from PC1(Dim1) and rest are from PC2(Dim2)

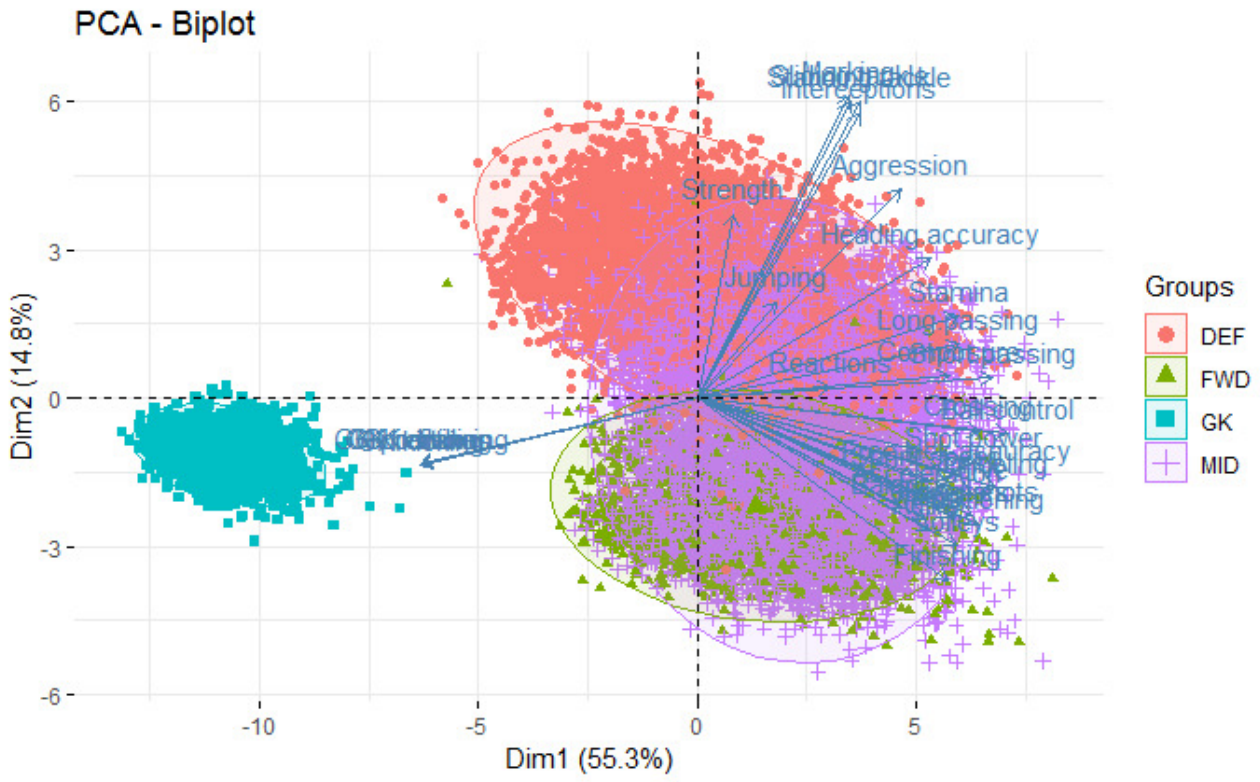
```
fviz_pca_biplot(pca_output)
```



biplots are graphs that provide a compact way of summarizing the relationships between individual and variables,

In this plot we can see all the variables associated with Goal Keeping have GK individuals clustered around them at left side of origin and rest all at the right with their variables.

```
fviz_pca_biplot(pca_output, geom.ind = "point", habillage = fifa1$Position, add
Ellipses = TRUE)
```



Creating ellipsoids based on the levels of the supplementary variable “Position”.

Creating ellipsoids based on the Nationality was not possible

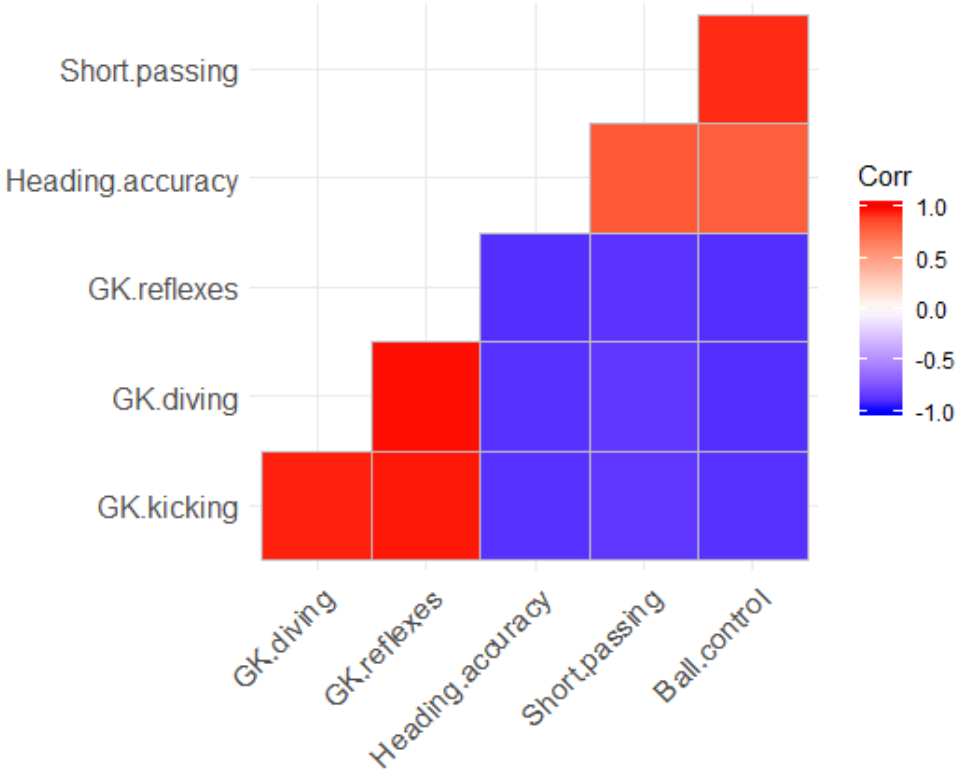
```
thirtyeight <- fifa %>% filter(Age >= 38) %>% select(Position,Short.passing,Hea
ding.accuracy,Ball.control,GK.kicking,GK.diving,GK.reflexes)
str(thirtyeight)
```

```
'data.frame':   72 obs. of  7 variables:
 $ Position        : Factor w/ 4 levels "DEF","FWD","GK",..: 3 3 4 3 1 3 1 3 4
4 ...
 $ Short.passing   : int  37 36 88 33 75 32 56 27 80 79 ...
 $ Heading.accuracy: int  13 10 39 12 79 21 76 10 33 58 ...
 $ Ball.control    : int  28 20 86 29 73 20 55 23 84 79 ...
 $ GK.kicking      : int  74 65 1 72 11 66 14 67 6 8 ...
 $ GK.diving       : int  89 80 5 76 12 78 6 74 7 11 ...
 $ GK.reflexes     : int  84 80 2 78 13 78 10 72 5 7 ...
```

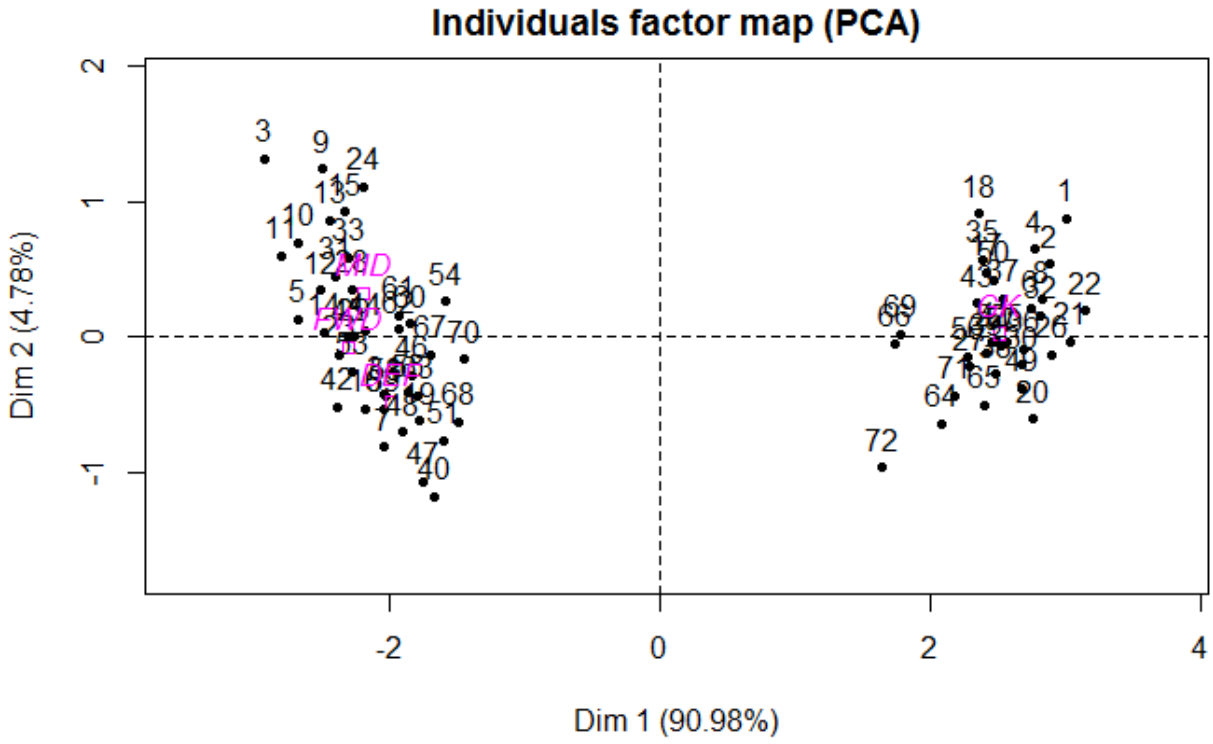
As in the first correlation matrix we saw that GK variables were negatively correlated with rest of the variables, i choosed short.passing,heading.accuracy,ball.control which are highly negatively correlated with GK variables to further analyZe

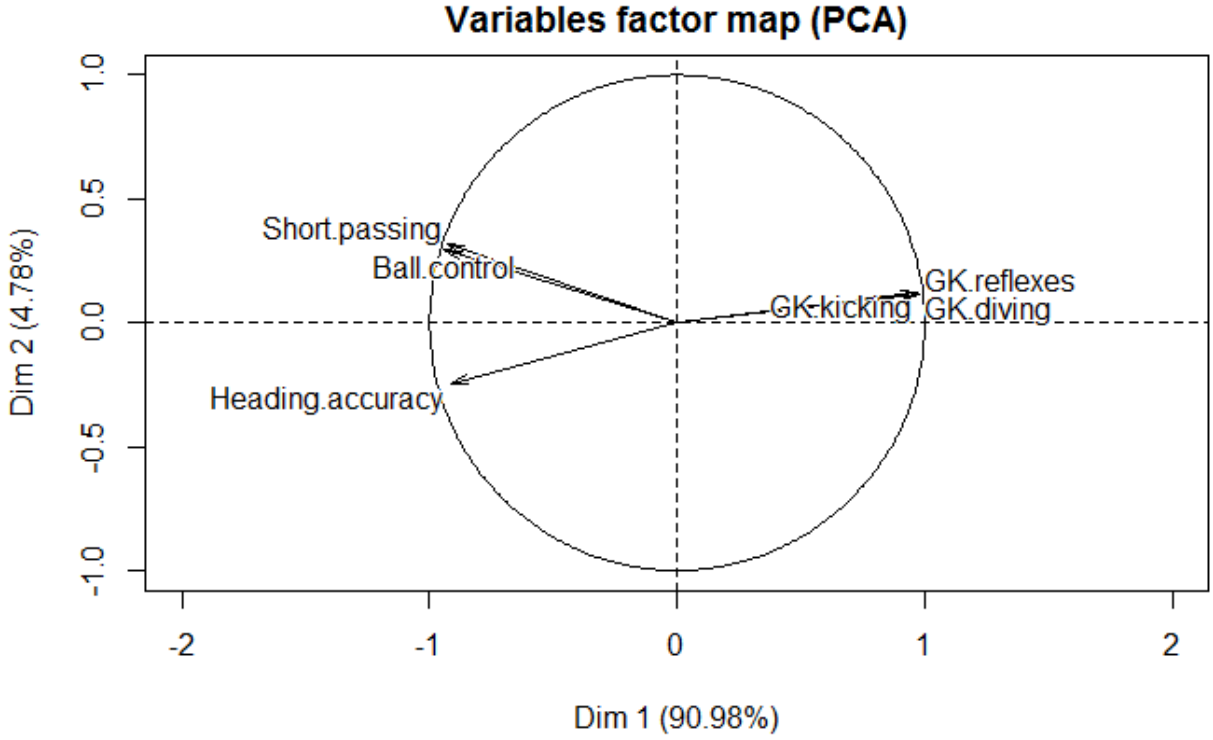
i am using a subset of data of players above age 38.

```
corr_38 <- cor(thirtyeight[2:7], use = "complete.obs")
ggcorrplot_clus <- ggcorrplot(corr_38, hc.order = TRUE, type = "lower")
ggcorrplot_clus
```




```
pca_f1 <- PCA(thirtyeight,quali.sup=1, scale.unit = TRUE, graph = TRUE)
```



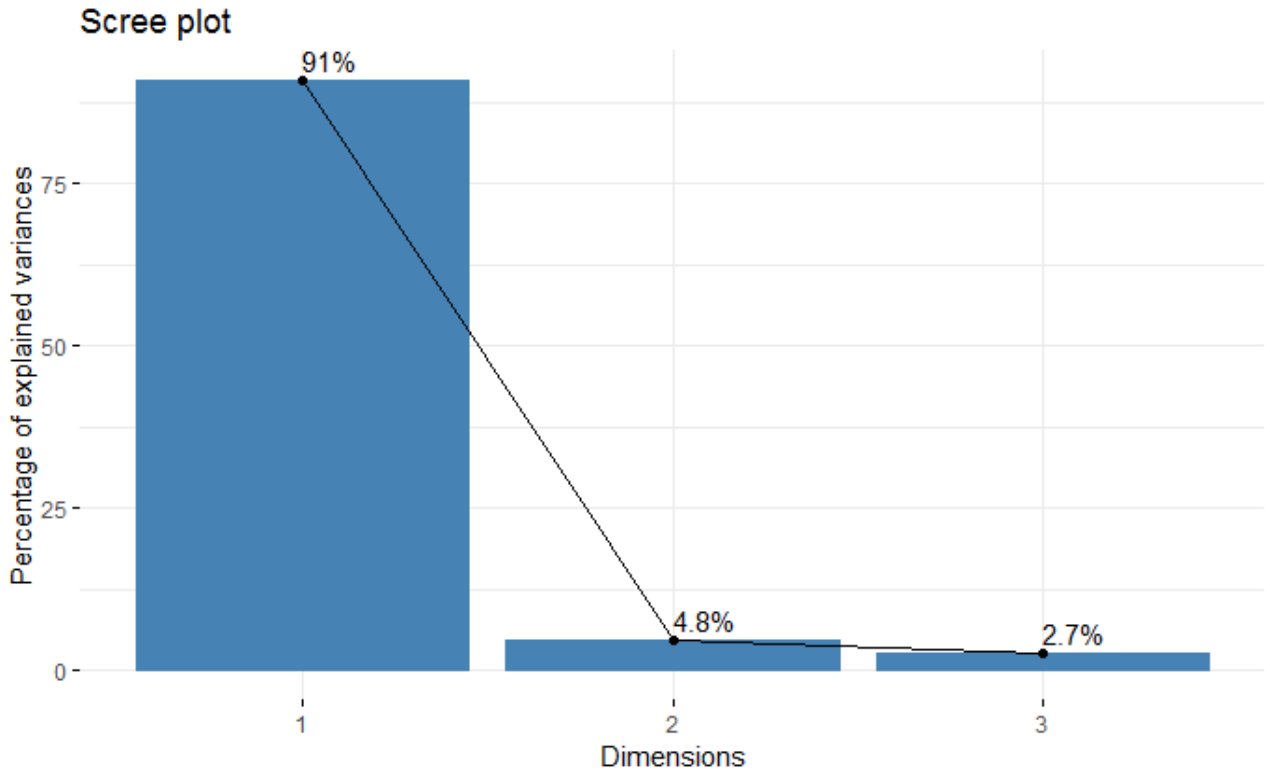


conducting principal components analysis on subset of data

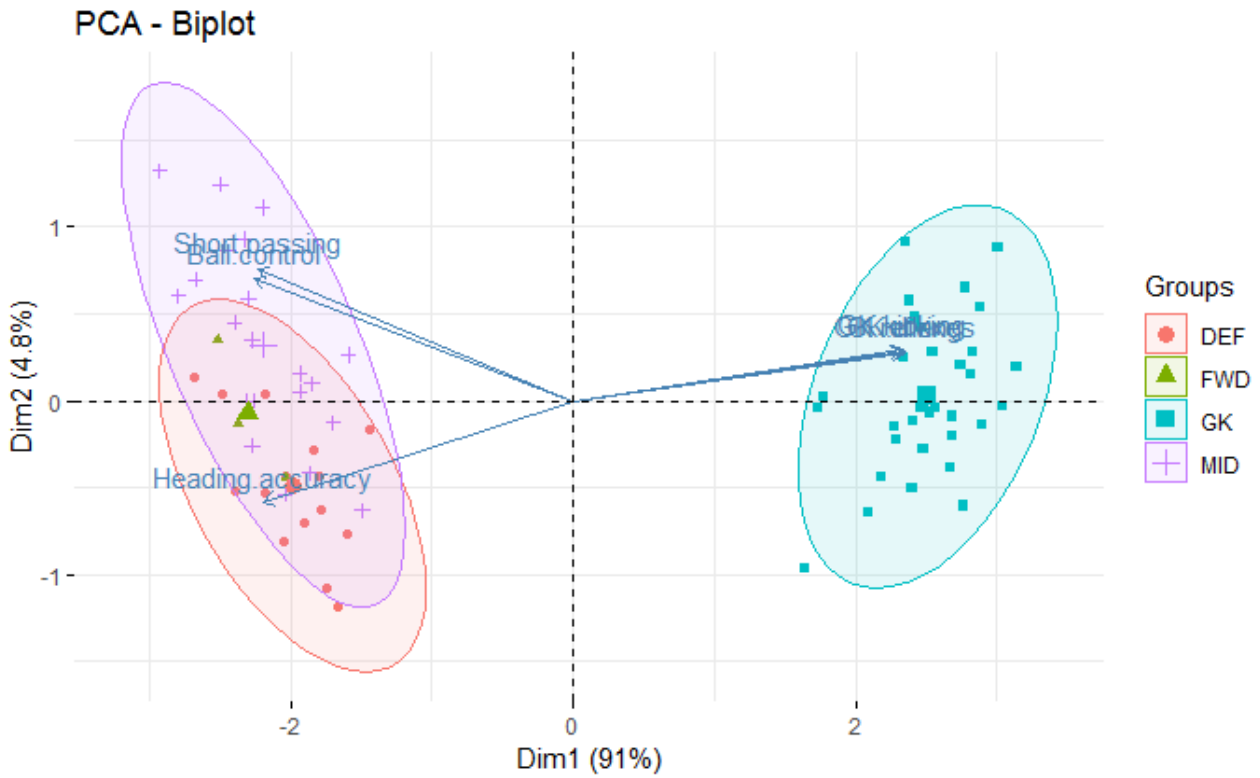
In the first plot, GK individuals 1 to 72 are plotted at right side and other skills individual at left.

PC1 has the variance of 90.98%, alone PC1 is more than enough

```
fviz_eig(pca_f1, addlabels = TRUE, ncp = 3)
```




```
fviz_pca_biplot(pca_f1, geom.ind = "point", habillage = thirtyeight$Position, a
ddEllipses = TRUE)
```



the variable “Position” is used as grouping variable.

references:

http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/ (http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/)

https://www.datacamp.com/courses/dimensionality-reduction-in-r (https://www.datacamp.com/courses/dimensionality-reduction-in-r)

https://www.youtube.com/watch?v=FgakZw6K1QQ (https://www.youtube.com/watch?v=FgakZw6K1QQ)