PCA

Maruf Ahmed Bhuiyan

7/31/2020

Principle Component Analysis (PCA)

Principal component analysis (PCA) allows us to summarize and to visualize the information in a data set containing individuals/observations described by multiple inter-correlated quantitative variables. Each variable could be considered as a different dimension. If you have more than 3 variables in your data sets, it could be very difficult to visualize a multi-dimensional hyperspace.

Principal component analysis is used to extract the important information from a multivariate data table and to express this information as a set of few new variables called principal components. These new variables correspond to a linear combination of the originals. The number of principal components is less than or equal to the number of original variables.

The information in a given data set corresponds to the total variation it contains. The goal of PCA is to identify directions (or principal components) along which the variation in the data is maximal. In other words, PCA reduces the dimensionality of a multivariate data to two or three principal components, that can be visualized graphically, with minimal loss of information. It is particularly helpful in the case of "wide" datasets, where you have many variables for each sample.

As you already read in the introduction, PCA is particularly handy when you're working with "wide" data sets. But why is that? Well, in such cases, where many variables are present, you cannot easily plot the data in its raw format, making it difficult to get a sense of the trends present within. PCA allows you to see the overall "shape" of the data, identifying which samples are similar to one another and which are very different. This can enable us to identify groups of samples that are similar and work out which variables make one group different from another.

Note that, the PCA method is particularly useful when the variables within the data set are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables (= principal components) explaining most of the variance in the original variables.

Caution: PCAs primary purpose is NOT as a ways of feature removal! PCA can reduce dimensionality but it wont reduce the number of features / variables in your data. What this means is that you might discover that you can explain 99% of variance in your 1000 feature dataset by just using 3 principal components but you still need those 1000 features to construct those 3 principal components, this also means that in the case of predicting on future data you still need those same 1000 features on your new observations to construct the corresponding principal components.

Eigenvector & Eigenvalue Just like many things in life, eigenvectors, and eigenvalues come in pairs: every eigenvector has a corresponding eigenvalue. Simply put, an eigenvector is a direction, such as "vertical" or "45 degrees", while an eigenvalue is a number telling you how much variance there is in the data in that direction. The eigenvector with the highest eigenvalue is, therefore, the first principal component.

So wait, there are possibly more eigenvalues and eigenvectors to be found in one data set?

That’s correct! The number of eigenvalues and eigenvectors that exits is equal to the number of dimensions the data set has. In the example that you saw above, there were 2 variables, so the data set was two-dimensional. That means that there are two eigenvectors and eigenvalues. Similarly, you’d find three pairs in a three-dimensional data set.

There are two general methods to perform PCA in R :
- Spectral decomposition which examines the covariances / correlations between variables
- Singular value decomposition which examines the covariances / correlations between individuals

The function *princomp()* uses the **spectral decomposition** approach. The functions *prcomp()* and PCA()[FactoMineR] use the **singular value decomposition (SVD)**.

prcomp() and princomp() functions The simplified format of these 2 functions are :

```
# prcomp(x, scale = FALSE)
# princomp(x, cor = FALSE, scores = TRUE)
```

1. Arguments for prcomp():

- x: a numeric matrix or data frame
- scale: a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place

2. Arguments for princomp():

- x: a numeric matrix or data frame
- cor: a logical value. If TRUE, the data will be centered and scaled before the analysis
- scores: a logical value. If TRUE, the coordinates on each principal component are calculated

The elements of the outputs returned by the functions prcomp() and princomp() includes :

prcomp()	princomp()	Description
sdev	sdev	the standard deviations of the principal components
rotation	loadings	the matrix of variable loadings (columns are eigenvectors)
center	center	the variable means (means that were substracted)
scale	scale	the variable standard deviations (the scaling applied to each variable)
x	scores	The coordinates of the individuals (observations) on the principal components.

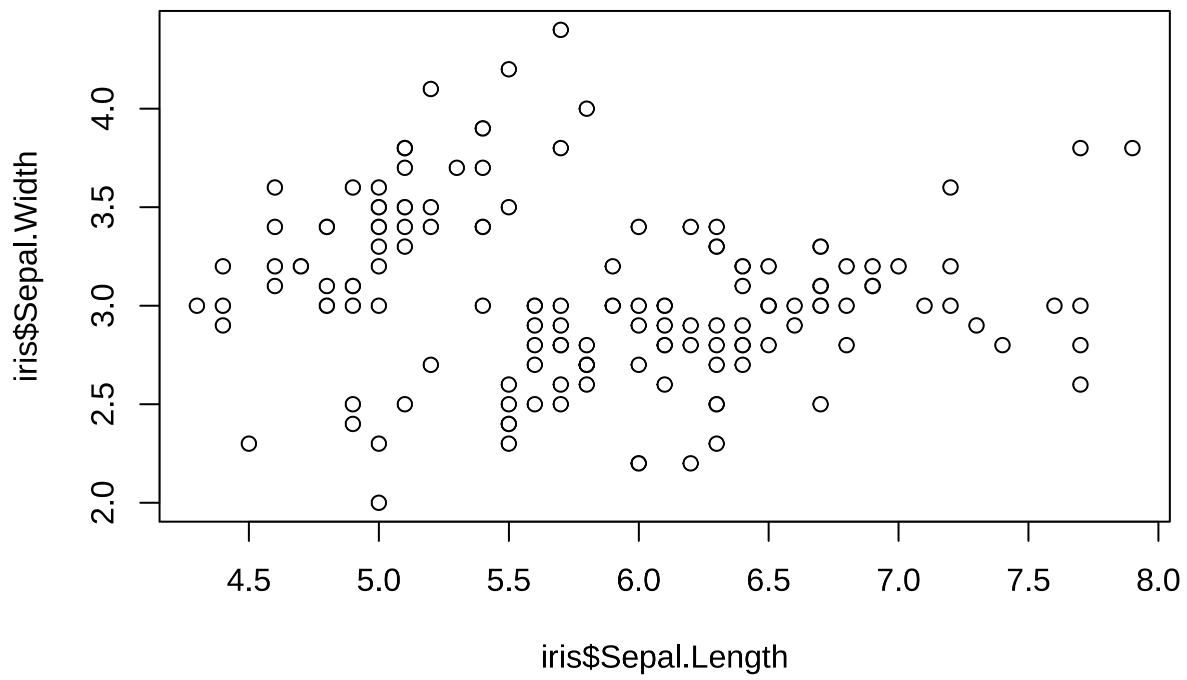
```r
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

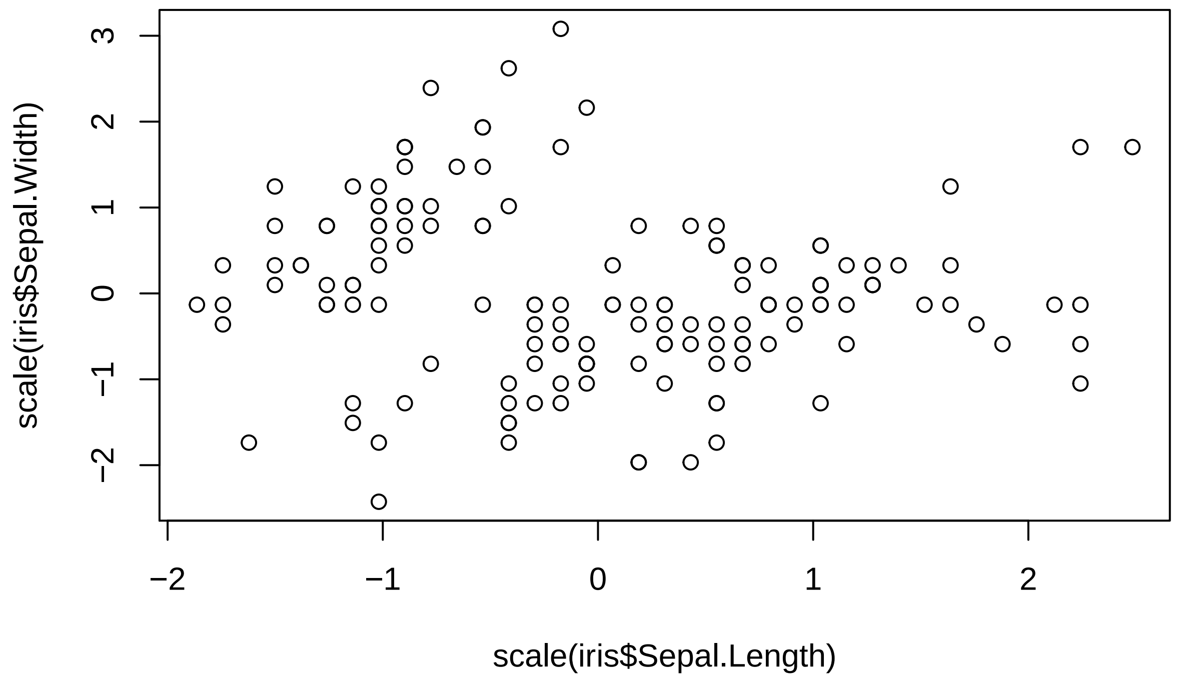
```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
##                 
```

```r
mypca <- prcomp(iris[,-5], scale = T)

plot(iris$Sepal.Length, iris$Sepal.Width)
```



```r
plot(scale(iris$Sepal.Length), scale(iris$Sepal.Width))
```



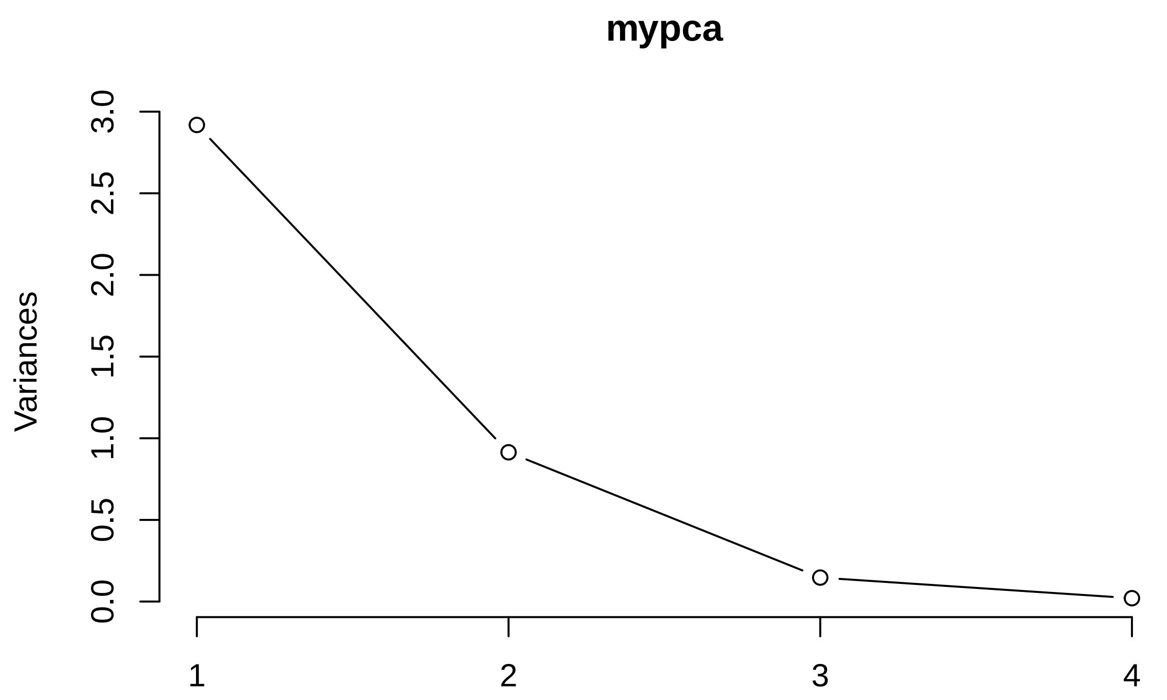
```
mypca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
summary(mypca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

```
# Standard deviation: This is simply the eigenvalues in our case
# since the data has been centered and scaled (standardized)

# Proportion of Variance: This is the amount of variance the
# component accounts for in the data, ie. PC1 accounts for >44%
# of total variance in the data alone!

# Cumulative Proportion: This is simply the accumulated amount
# of explained variance, ie. if we used the first 10 components
# we would be able to account for >95% of total variance in the data.

plot(mypca, type = "l")
```



```
biplot(mypca, scale = 0)

# Extract PCA score
str(mypca)
```

```
## List of 5
##  $ sdev    : num [1:4] 1.708 0.956 0.383 0.144
##  $ rotation: num [1:4, 1:4] 0.521 -0.269 0.58 0.565 -0.377 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##   .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
##  $ center  : Named num [1:4] 5.84 3.06 3.76 1.2
##   ..- attr(*, "names")= chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##  $ scale   : Named num [1:4] 0.828 0.436 1.765 0.762
##   ..- attr(*, "names")= chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##  $ x       : num [1:150, 1:4] -2.26 -2.07 -2.36 -2.29 -2.38 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
##  - attr(*, "class")= chr "prcomp"
```

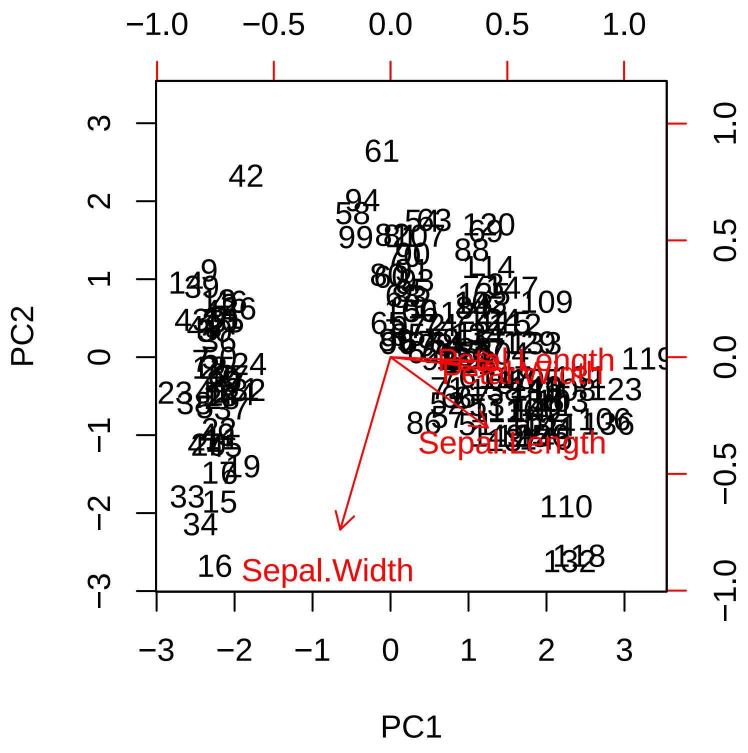
```
head(mypca$x)
```

```
##            PC1        PC2         PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

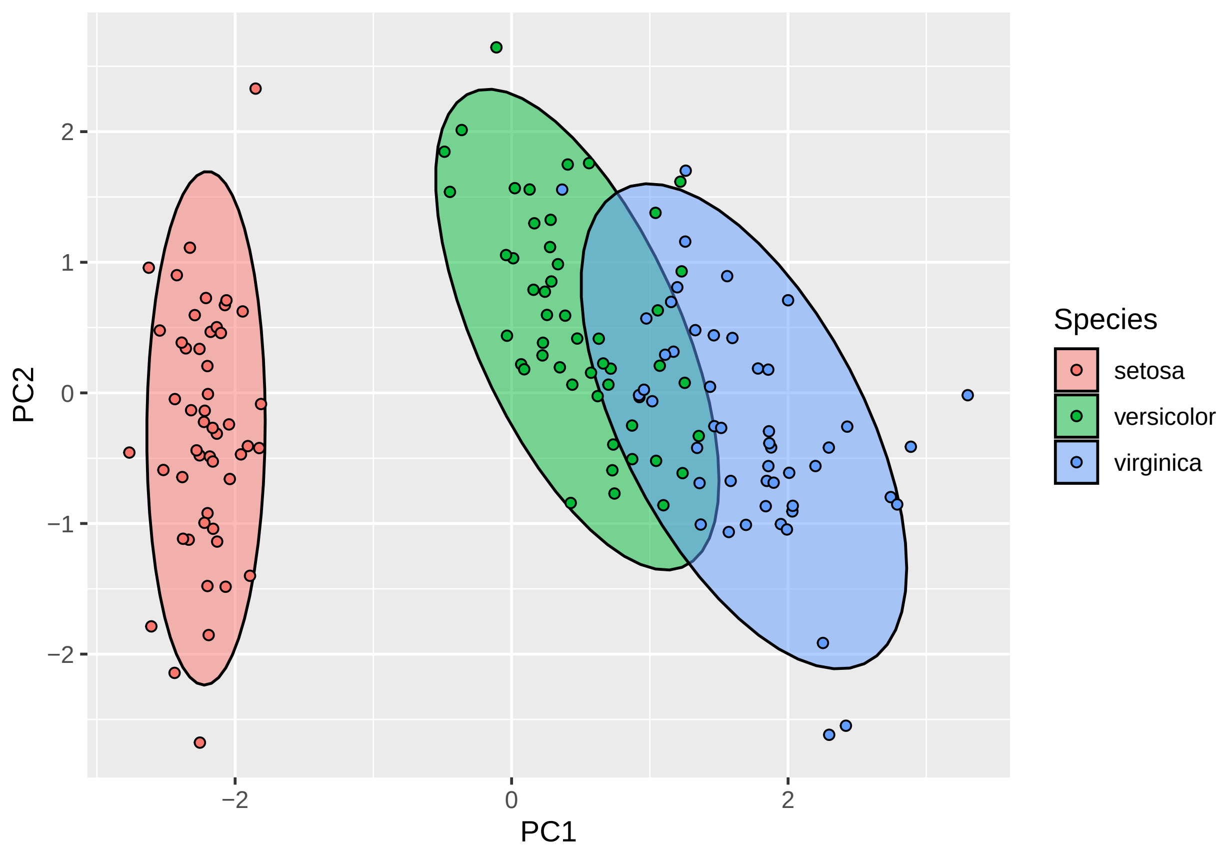
```
iris2 <- cbind(iris, mypca$x[,1:2])
head(iris2)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species       PC1
## 1          5.1         3.5          1.4         0.2  setosa -2.257141
## 2          4.9         3.0          1.4         0.2  setosa -2.074013
## 3          4.7         3.2          1.3         0.2  setosa -2.356335
## 4          4.6         3.1          1.5         0.2  setosa -2.291707
## 5          5.0         3.6          1.4         0.2  setosa -2.381863
## 6          5.4         3.9          1.7         0.4  setosa -2.068701
##          PC2
## 1 -0.4784238
## 2  0.6718827
## 3  0.3407664
## 4  0.5953999
## 5 -0.6446757
## 6 -1.4842053
```

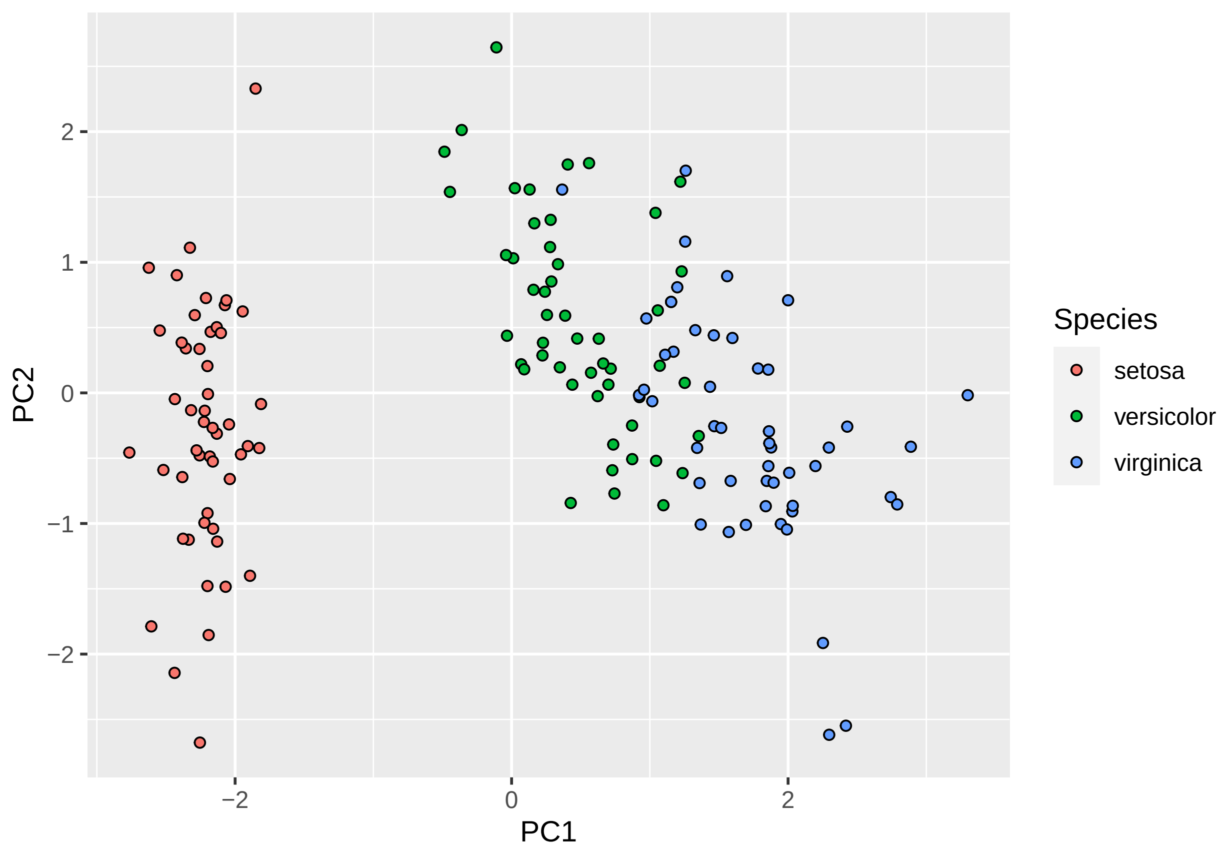
```
# Plot with ggplot
library(ggplot2)
```



```r
ggplot(iris2, aes(PC1, PC2, col = Species, fill = Species)) +
  stat_ellipse(geom = "polygon", col = "black", alpha = 0.5) +
  geom_point(shape = 21, col = "black")
```



```
ggplot(iris2, aes(PC1, PC2, col = Species, fill = Species)) +
  geom_point(shape = 21, col = "black")
```



```r
# Correlation with variables and PCA

cor(iris[, -5], iris2[, 6:7])
```

```
##                     PC1         PC2
## Sepal.Length  0.8901688 -0.36082989
## Sepal.Width  -0.4601427 -0.88271627
## Petal.Length  0.9915552 -0.02341519
## Petal.Width   0.9649790 -0.06399985
```

PCA by DataCamp

URL: https://www.datacamp.com/community/tutorials/pca-analysis-r

A Simple PCA In this section, you will try a PCA using a simple and easy to understand dataset. You will use the mtcars dataset, which is built into R. This dataset consists of data on 32 models of car, taken from an American motoring magazine (1974 Motor Trend magazine). For each car, you have 11 features, expressed in varying units (US units).

Because PCA works best with numerical data, you'll exclude the two categorical variables (vs and am).

We are left with a matrix of 9 columns and 32 rows, which you pass to the prcomp() function, assigning your output to mtcars.pca. We will also set two arguments, center and scale, to be TRUE. Then you can have a peek at your PCA object with summary().

```r
data(mtcars)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
#PCA is sensitive to variances of values. So, we have
# to convert standardize the values using scale.
# It expresses the values in terms of their sd.
pca_mtcars <- prcomp(mtcars[,c(1:7, 10, 11)], scale = T, center = T)

summary(pca_mtcars)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.3782 1.4429 0.71008 0.51481 0.42797 0.35184 0.32413
## Proportion of Variance 0.6284 0.2313 0.05602 0.02945 0.02035 0.01375 0.01167
## Cumulative Proportion  0.6284 0.8598 0.91581 0.94525 0.96560 0.97936 0.99103
##                           PC8     PC9
## Standard deviation     0.2419 0.14896
## Proportion of Variance 0.0065 0.00247
## Cumulative Proportion  0.9975 1.00000
```

```r
# Principle Components are ordered according to the significance.
# PC1 & PC2 combined can describe 86% (cumulative prop) of variance in the dataset.
```

We get 9 principal components, which you call PC1-9. Each of these explains a percentage of the total variation in the dataset. That is to say: PC1 explains 63% of the total variance, which means that nearly two-thirds of the information in the dataset (9 variables) can be encapsulated by just that one Principal Component. PC2 explains 23% of the variance. So, by knowing the position of a sample in relation to just PC1 and PC2, we can get a very accurate view on where it stands in relation to other samples, as just PC1 and PC2 can explain 86% of the variance.

```r
pca_mtcars
```

```
## Standard deviations (1, .., p=9):
## [1] 2.3782219 1.4429485 0.7100809 0.5148082 0.4279704 0.3518426 0.3241326
## [8] 0.2418962 0.1489644
##
## Rotation (n x k) = (9 x 9):
##             PC1         PC2         PC3          PC4        PC5         PC6
## mpg  -0.3931477  0.02753861 -0.22119309 -0.006126378 -0.3207620  0.72015586
## cyl   0.4025537  0.01570975 -0.25231615  0.040700251  0.1171397  0.22432550
## disp  0.3973528 -0.08888469 -0.07825139  0.339493732 -0.4867849 -0.01967516
## hp    0.3670814  0.26941371 -0.01721159  0.068300993 -0.2947317  0.35394225
## drat -0.3118165  0.34165268  0.14995507  0.845658485  0.1619259 -0.01536794
## wt    0.3734771 -0.17194306  0.45373418  0.191260029 -0.1874822 -0.08377237
## qsec -0.2243508 -0.48404435  0.62812782 -0.030329127 -0.1482495  0.25752940
## gear -0.2094749  0.55078264  0.20658376 -0.282381831 -0.5624860 -0.32298239
## carb  0.2445807  0.48431310  0.46412069 -0.214492216  0.3997820  0.35706914
##              PC7         PC8         PC9
## mpg  -0.38138068 -0.12465987  0.11492862
## cyl  -0.15893251  0.81032177  0.16266295
## disp -0.18233095 -0.06416707 -0.66190812
## hp    0.69620751 -0.16573993  0.25177306
## drat  0.04767957  0.13505066  0.03809096
## wt   -0.42777608 -0.19839375  0.56918844
## qsec  0.27622581  0.35613350 -0.16873731
## gear -0.08555707  0.31636479  0.04719694
## carb -0.20604210 -0.10832772 -0.32045892
```

```r
str(pca_mtcars)
```

```
## List of 5
##  $ sdev    : num [1:9] 2.378 1.443 0.71 0.515 0.428 ...
##  $ rotation: num [1:9, 1:9] -0.393 0.403 0.397 0.367 -0.312 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:9] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:9] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:9] 20.09 6.19 230.72 146.69 3.6 ...
##   ..- attr(*, "names")= chr [1:9] "mpg" "cyl" "disp" "hp" ...
##  $ scale   : Named num [1:9] 6.027 1.786 123.939 68.563 0.535 ...
##   ..- attr(*, "names")= chr [1:9] "mpg" "cyl" "disp" "hp" ...
##  $ x       : num [1:32, 1:9] -0.664 -0.637 -2.3 -0.215 1.587 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##   .. ..$ : chr [1:9] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

Our PCA object contains the following information:

- The center point (center), scaling (scale), standard deviation(sdev) of each principal component
- The relationship (correlation or anticorrelation, etc) between the initial variables and the principal components (rotation)
- The values of each sample in terms of the principal components (x)

Plotting PCA Now it's time to plot our PCA. You will make a biplot, which includes both the position of each sample in terms of PC1 and PC2 and also will show us how the initial variables map onto this. We will use the ggbiplot package, which offers a user-friendly and pretty function to plot biplots. A biplot is a type of plot that will allow you to visualize how the samples relate to one another in our PCA (which samples are similar and which are different) and will simultaneously reveal how each variable contributes to each principal component.

Before we get started, let's first install ggbiplot!

```r
library(devtools)
```

```
## Loading required package: usethis
```

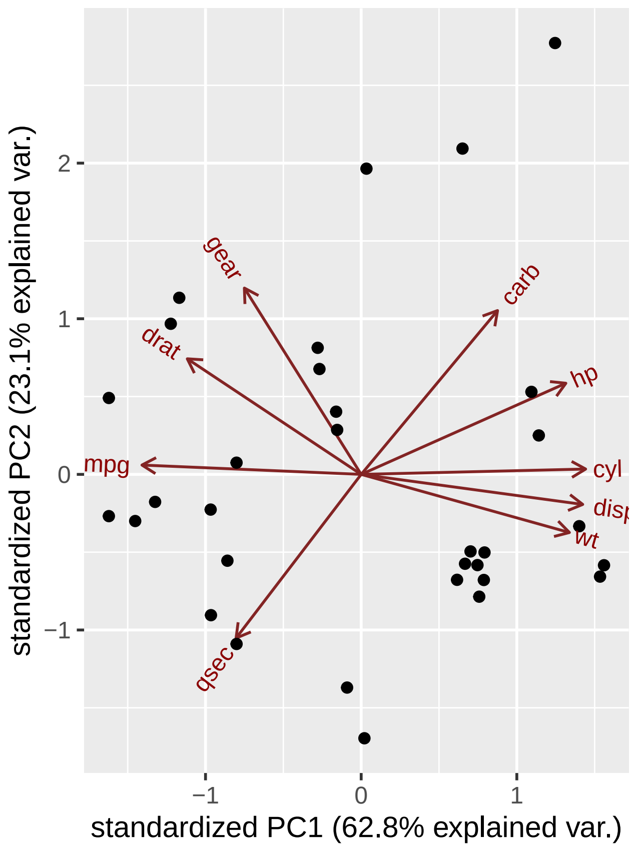
```r
# install_github("vqv/ggbiplot")
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

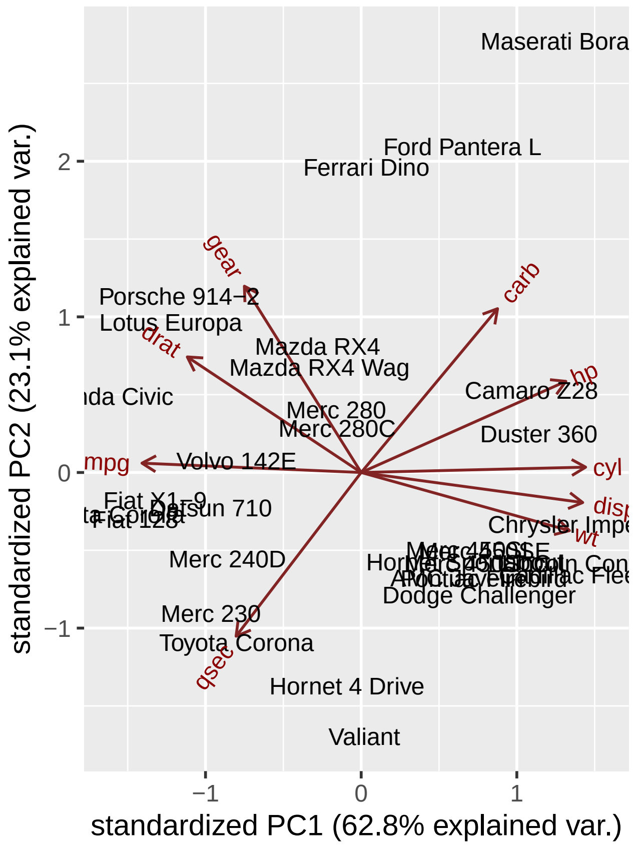
```r
ggbiplot(pca_mtcars)
```



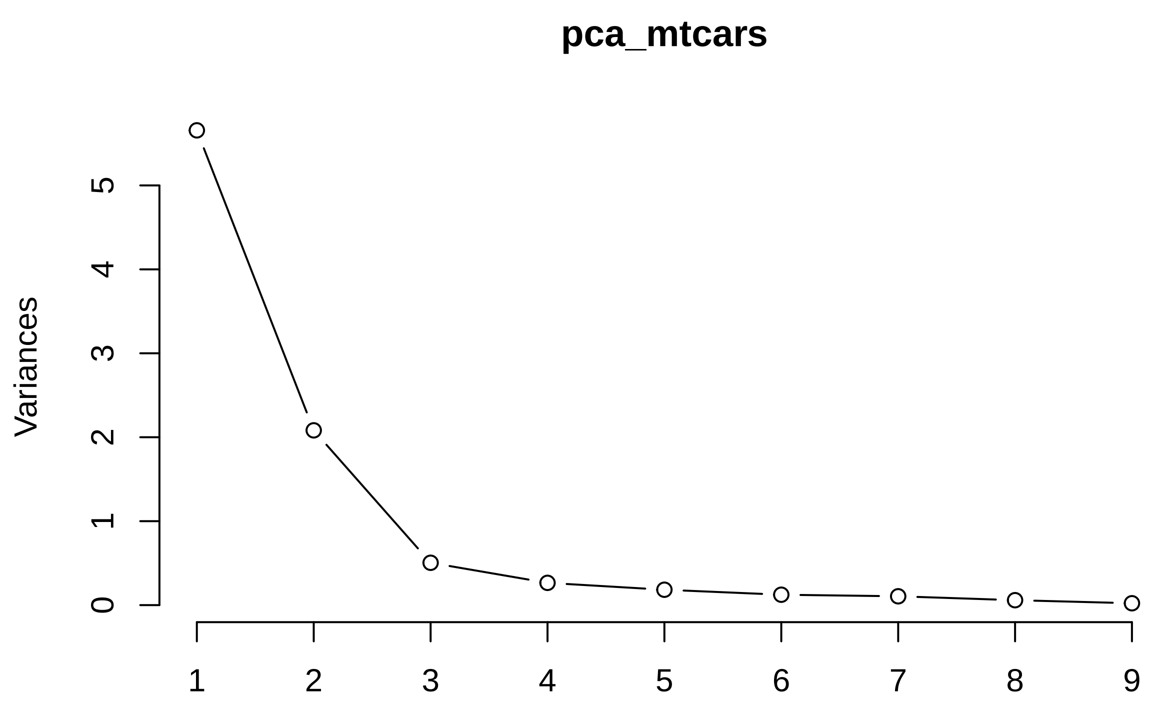
The axes are seen as arrows originating from the center point. Here, you see that the variables hp, cyl, and disp all contribute to PC1, with higher values in those variables moving the samples to the right on this plot. This lets you see how the data points relate to the axes, but it's not very informative without knowing which point corresponds to which sample (car).

You'll provide an argument to ggbiplot: let's give it the rownames of mtcars as labels. This will name each point with the name of the car in question:

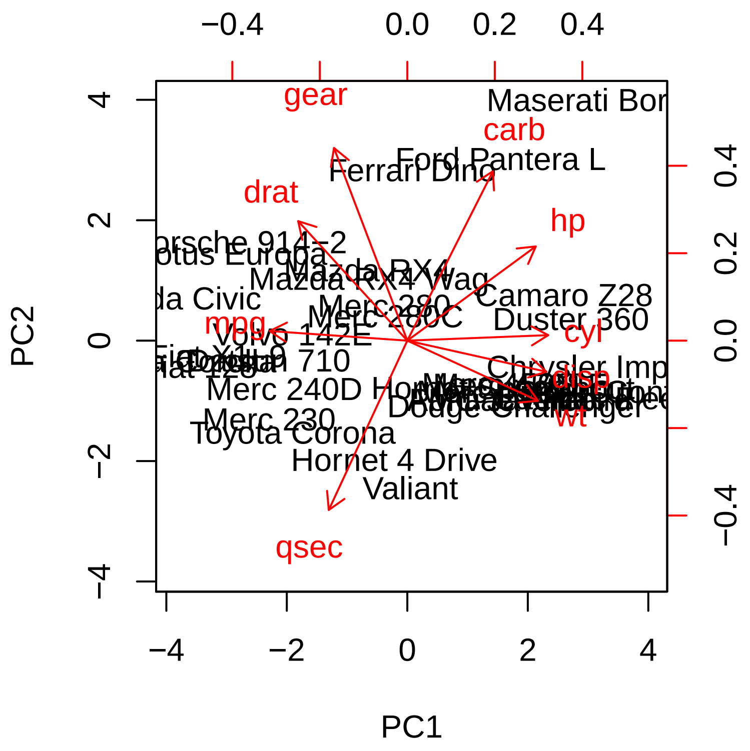
```r
ggbiplot(pca_mtcars, labels=rownames(mtcars))
```



```r
# we can also do this using base R
plot(pca_mtcars, type = "l")  # this is a scree plot
```



```
# biplot
biplot(pca_mtcars, scale = 0)
```



Now you can see which cars are similar to one another. For example, the Maserati Bora, Ferrari Dino and Ford Pantera L all cluster together at the top. This makes sense, as all of these are sports cars.

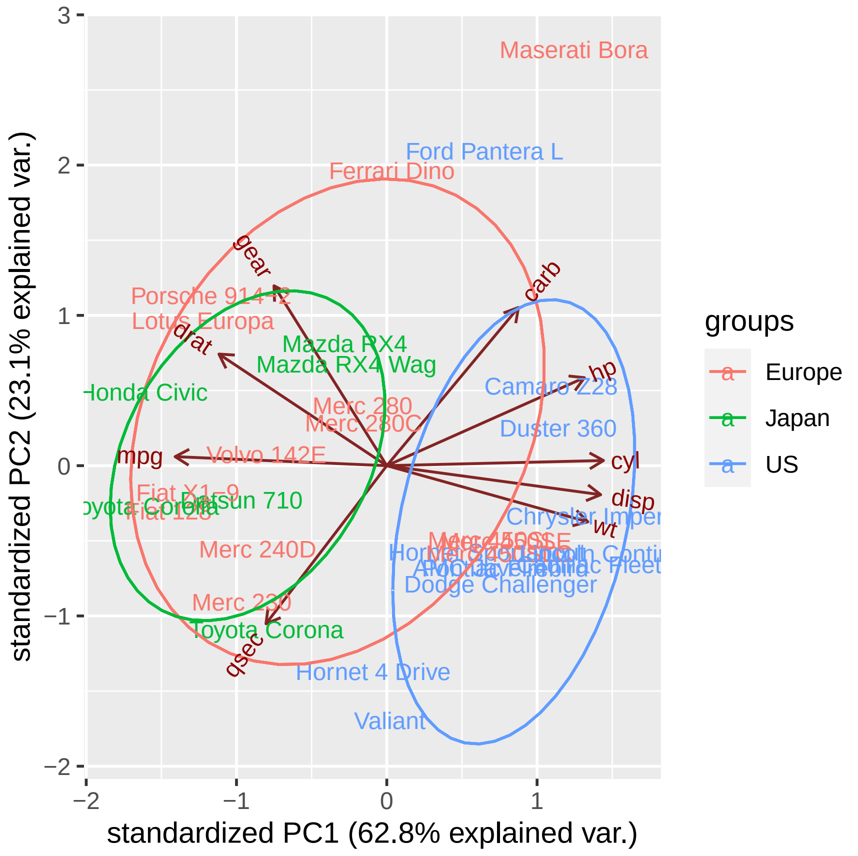
How else can you try to better understand your data?

Interpreting the results Maybe if we look at the origin of each of the cars. We'll put them into one of three categories (categories?), one each for the US, Japanese and European cars. You make a list for this info, then pass it to the groups argument of ggbiplot. You'll also set the ellipse argument to be TRUE, which will draw an ellipse around each group.

```r
# Let's group the cars according to their country of origin
rownames(mtcars)
```

```
##  [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
##  [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
##  [7] "Duster 360"          "Merc 240D"           "Merc 230"
## [10] "Merc 280"            "Merc 280C"           "Merc 450SE"
## [13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetwood"
## [16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
## [19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
## [22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
## [25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
## [28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
## [31] "Maserati Bora"       "Volvo 142E"
```

```r
# First the cars are form Japan, and so on.
mtcars_country <- c(rep("Japan", 3), rep("US",4),
                    rep("Europe", 7), rep("US",3),
                    "Europe", rep("Japan", 3),
                    rep("US",4), rep("Europe", 3),
                    "US", rep("Europe", 3))

ggbiplot(pca_mtcars, ellipse=TRUE, labels=rownames(mtcars),
         groups= mtcars_country)
```

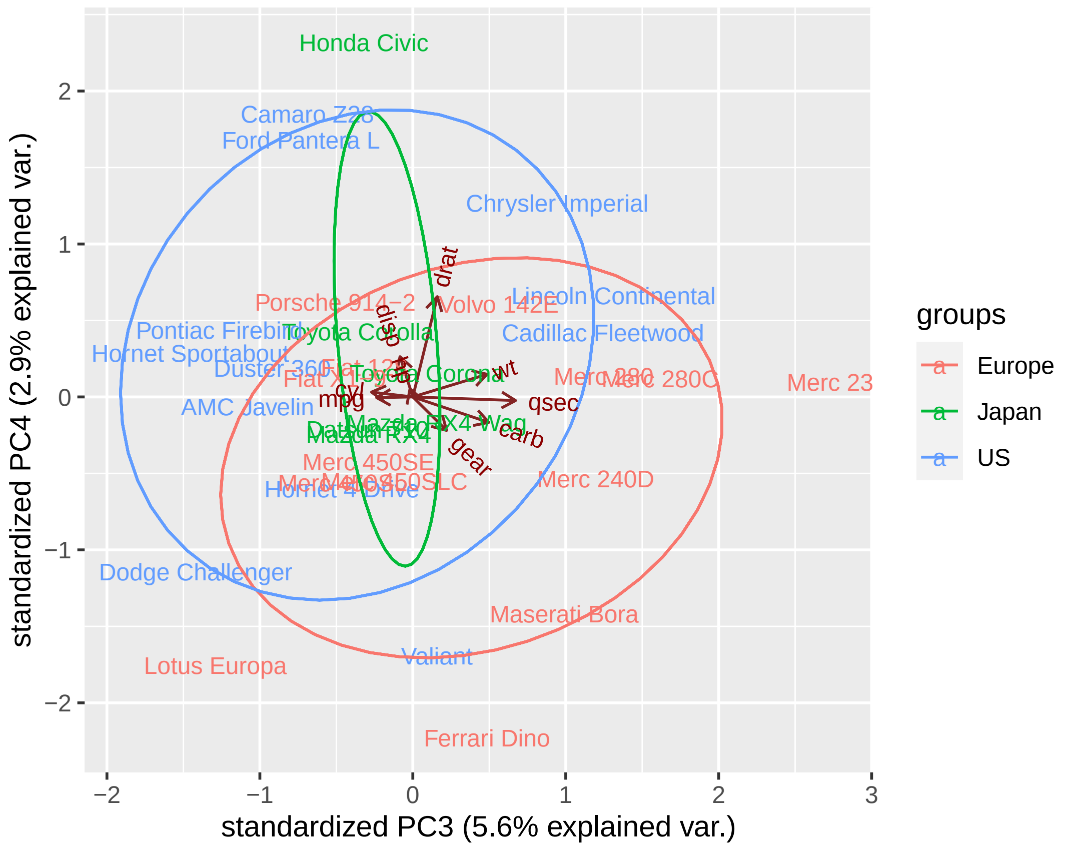


Now you see something interesting: the American cars form a distinct cluster to the right. Looking at the axes, you see that the American cars are characterized by high values for cyl, disp, and wt. Japanese cars, on the other hand, are characterized by high mpg. European cars are somewhat in the middle and less tightly clustered than either group.

Of course, you have many principal components available, each of which map differently to the original variables. You can also ask ggbiplot to plot these other components, by using the choices argument.

Let's have a look at PC3 and PC4:

```r
ggbiplot(pca_mtcars, ellipse = T, choices = c(3,4),
         labels = rownames(mtcars), groups= mtcars_country)
```

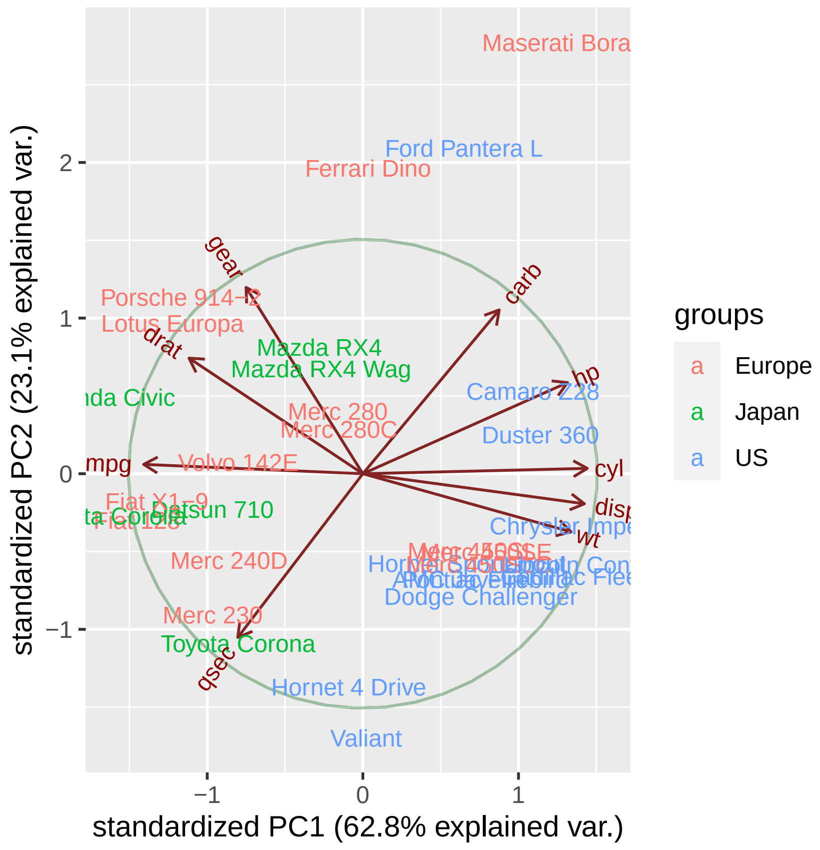


You don't see much here, but this isn't too surprising. PC3 and PC4 explain very small percentages of the total variation, so it would be surprising if you found that they were very informative and separated the groups or revealed apparent patterns.

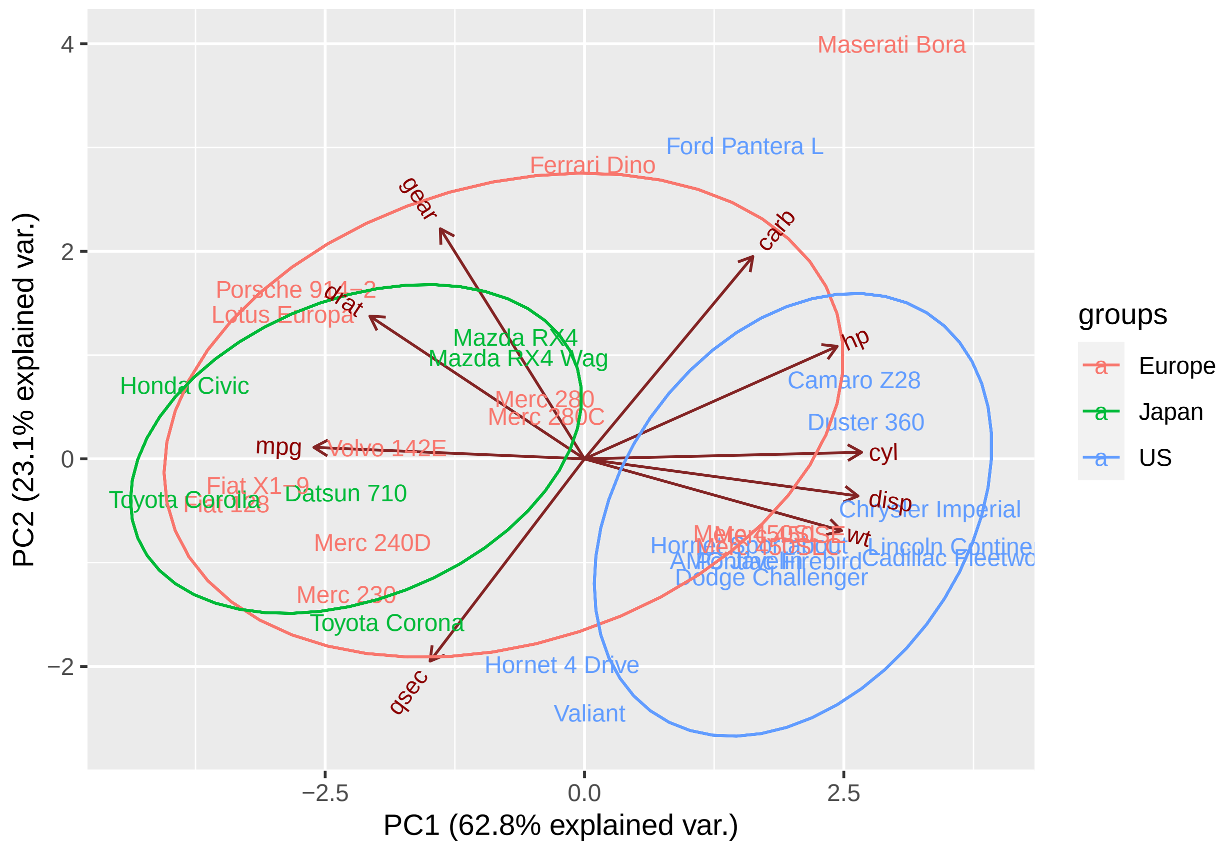
Let's take a moment to recap: having performed a PCA using the mtcars dataset, we can see a clear separation between American and Japanese cars along a principal component that is closely correlated to cyl, disp, wt, and mpg. This provides us with some clues for future analyses; if we were to try to build a classification model to identify the origin of a car, these variables might be useful.

Graphical parameters with ggbiplot There are also some other variables we can play with to alter your biplots. We can add a circle to the center of the dataset (circle argument):

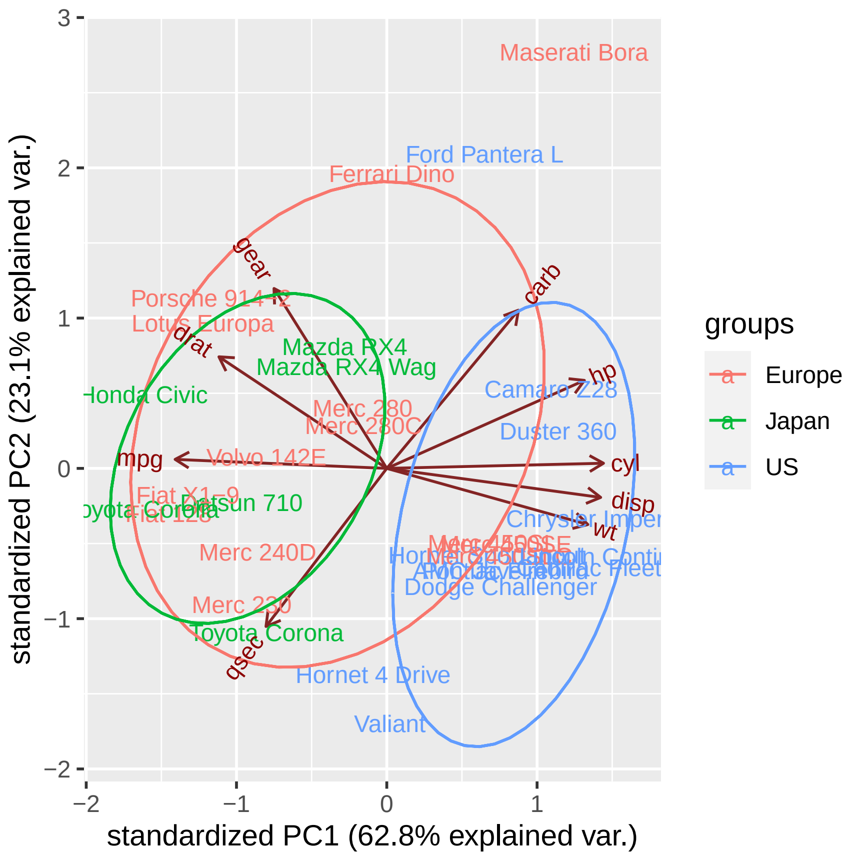
```r
ggbiplot(pca_mtcars, circle = T, labels = rownames(mtcars),
         groups = mtcars_country)
```



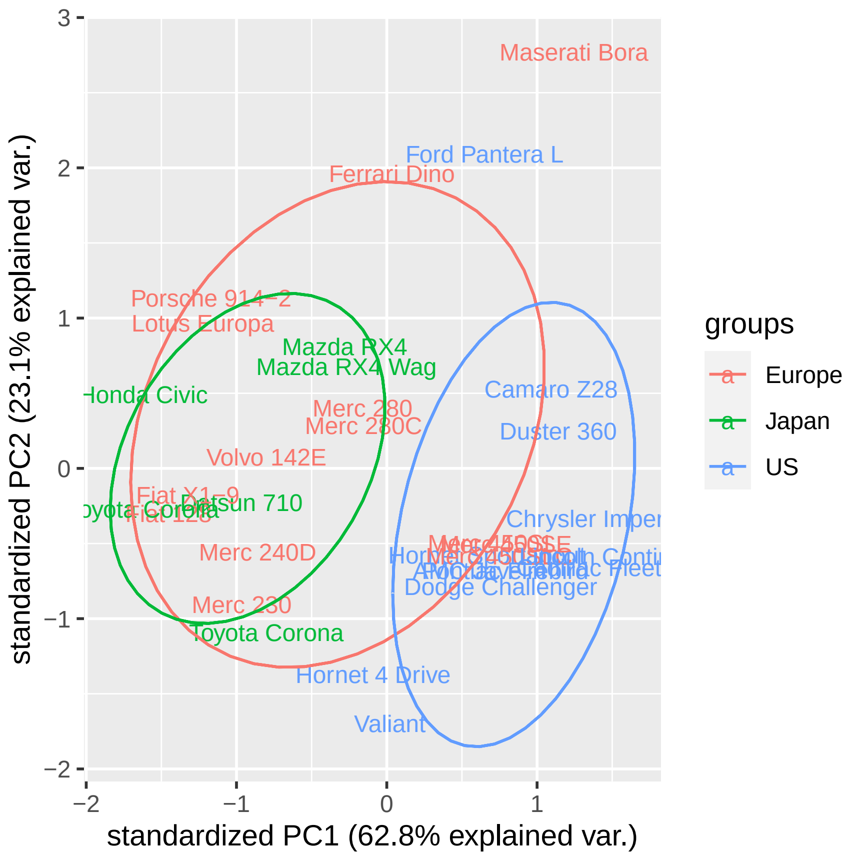
```r
# We can also scale the samples (obs.scale) and the variables (var.scale):
ggbiplot(pca_mtcars, ellipse = T, obs.scale = 1, var.scale = 1,
         labels = rownames(mtcars), groups=mtcars_country)
```



```r
# Let's see a plot without scaling
ggbiplot(pca_mtcars, ellipse = T, labels=rownames(mtcars),
         groups= mtcars_country)
```



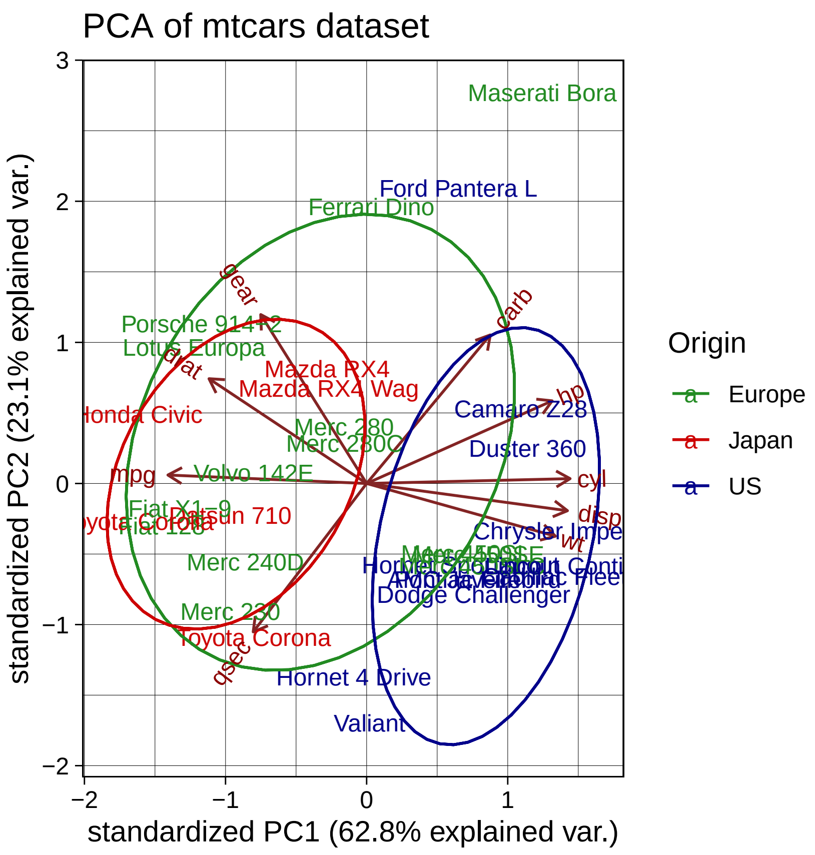
```r
# We can also remove the arrows altogether, using var.axes.
ggbiplot(pca_mtcars, ellipse = T, var.axes = F,
         labels = rownames(mtcars), groups = mtcars_country)
```



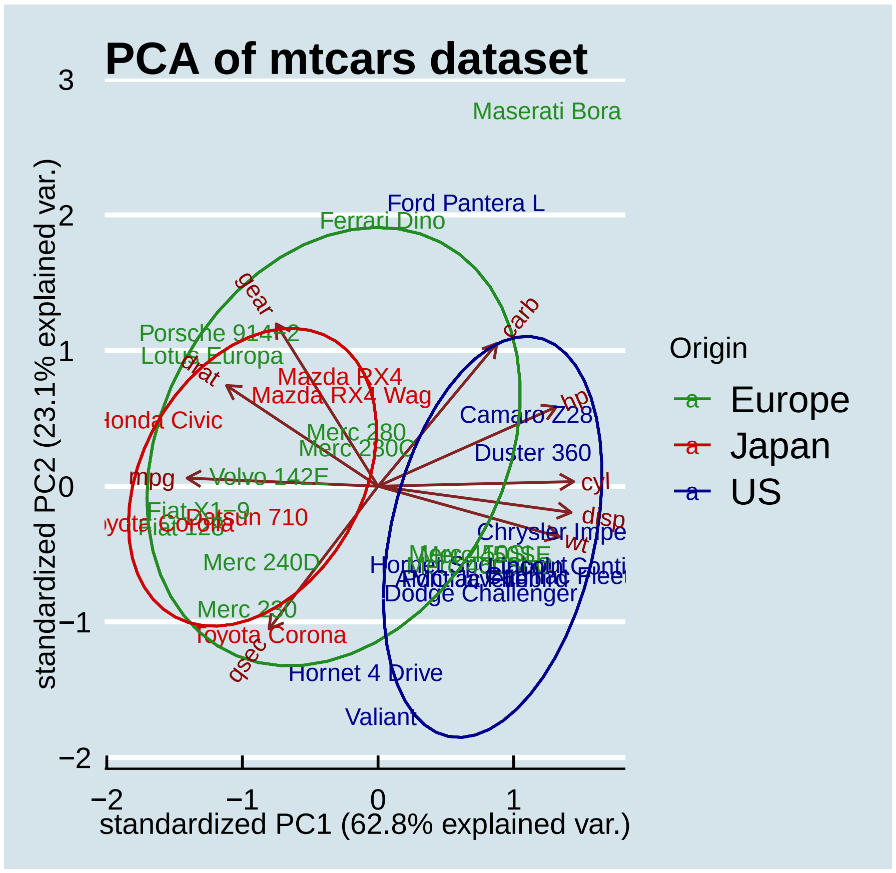
Customize ggbiplot As ggbiplot is based on the ggplot function, you can use the same set of graphical parameters to alter your biplots as you would for any ggplot. Here, you're going to:

- Specify the colours to use for the groups with scale_colour_manual()
- Add a title with ggtitle()
- Specify the minimal() theme
- Move the legend with theme()

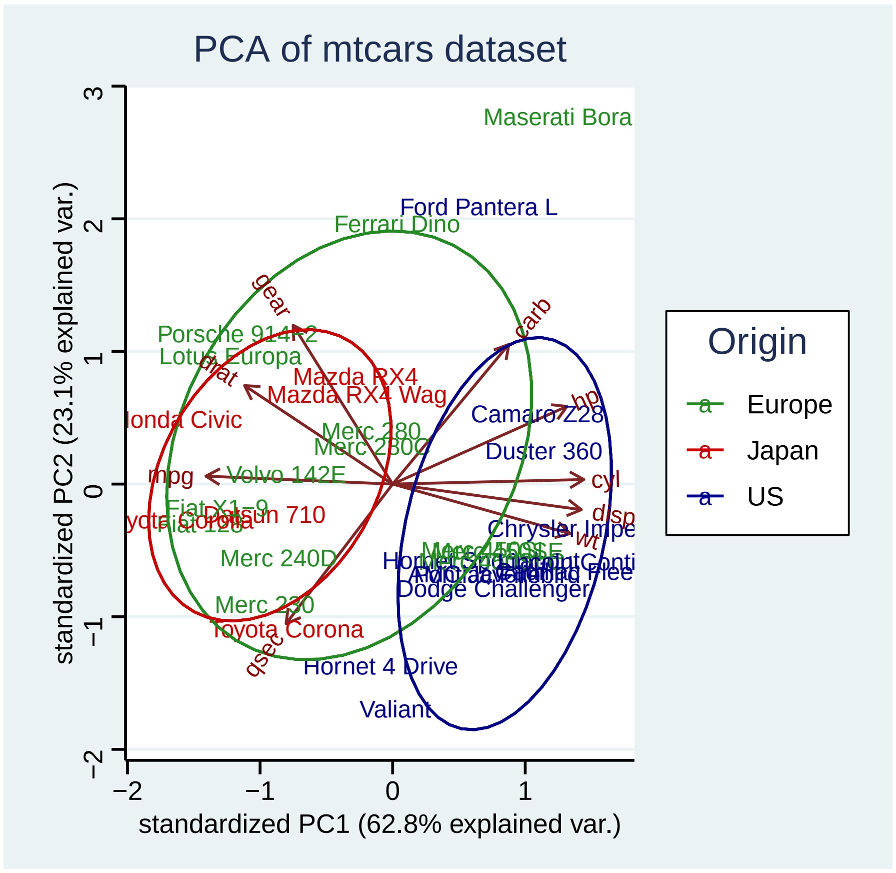
```
ggbiplot(pca_mtcars, ellipse = T, labels=rownames(mtcars),
         groups= mtcars_country) +
  scale_colour_manual(name="Origin",
                      values= c("forest green", "red3", "dark blue"))+
  ggtitle("PCA of mtcars dataset") +
  theme_linedraw() +
  theme(legend.position = "right")
```



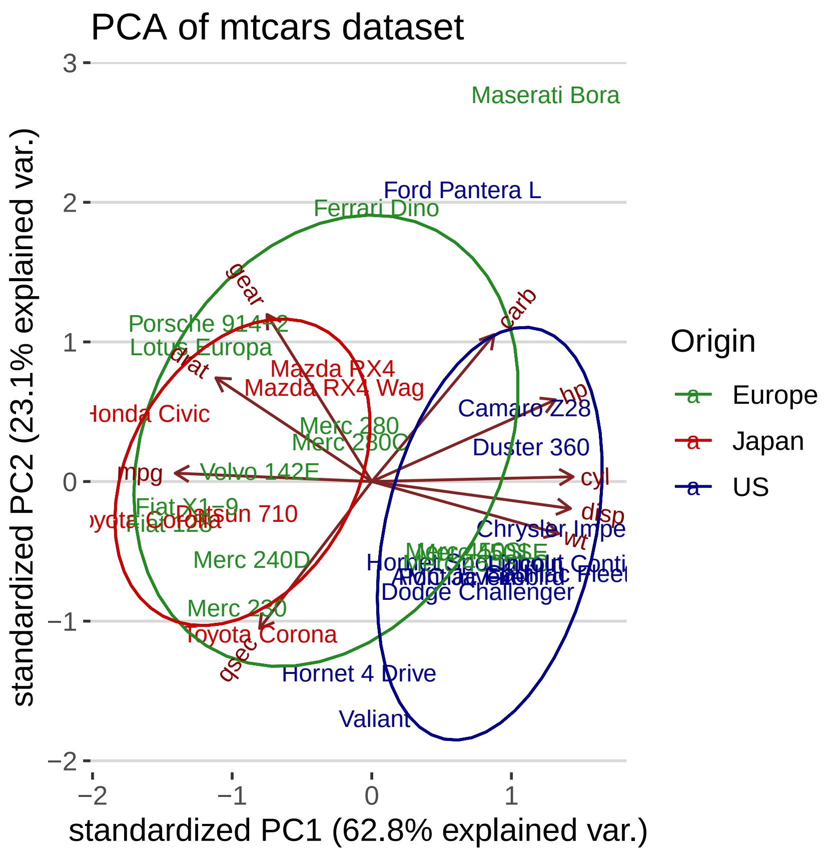
```r
# install.packages("ggthemes")
library(ggthemes)

# theme_economist
ggbiplot(pca_mtcars, ellipse = T, labels=rownames(mtcars),
         groups= mtcars_country) +
  scale_colour_manual(name="Origin",
                      values= c("forest green", "red3", "dark blue"))+
  ggtitle("PCA of mtcars dataset") +
  theme_economist() +
  scale_fill_economist() +
  theme(legend.position = "right")
```



```r
# theme_stata
ggbiplot(pca_mtcars, ellipse = T, labels=rownames(mtcars),
         groups= mtcars_country) +
  scale_colour_manual(name="Origin",
                      values= c("forest green", "red3", "dark blue"))+
  ggtitle("PCA of mtcars dataset") +
  theme_stata() +
  scale_fill_stata() +
  theme(legend.position = "right")
```



```r
# theme_hc
ggbiplot(pca_mtcars, ellipse = T, labels=rownames(mtcars),
         groups= mtcars_country) +
  scale_colour_manual(name="Origin",
                      values= c("forest green", "red3", "dark blue"))+
  ggtitle("PCA of mtcars dataset") +
  theme_hc() +
  theme(legend.position = "right")
```



Adding a new sample

Okay, so let's say you want to add a new sample to your dataset. This is a very special car, with stats unlike any other. It's super-powerful, has a 60-cylinder engine, amazing fuel economy, no gears and is very light. It's a "spacecar", from Jupiter.

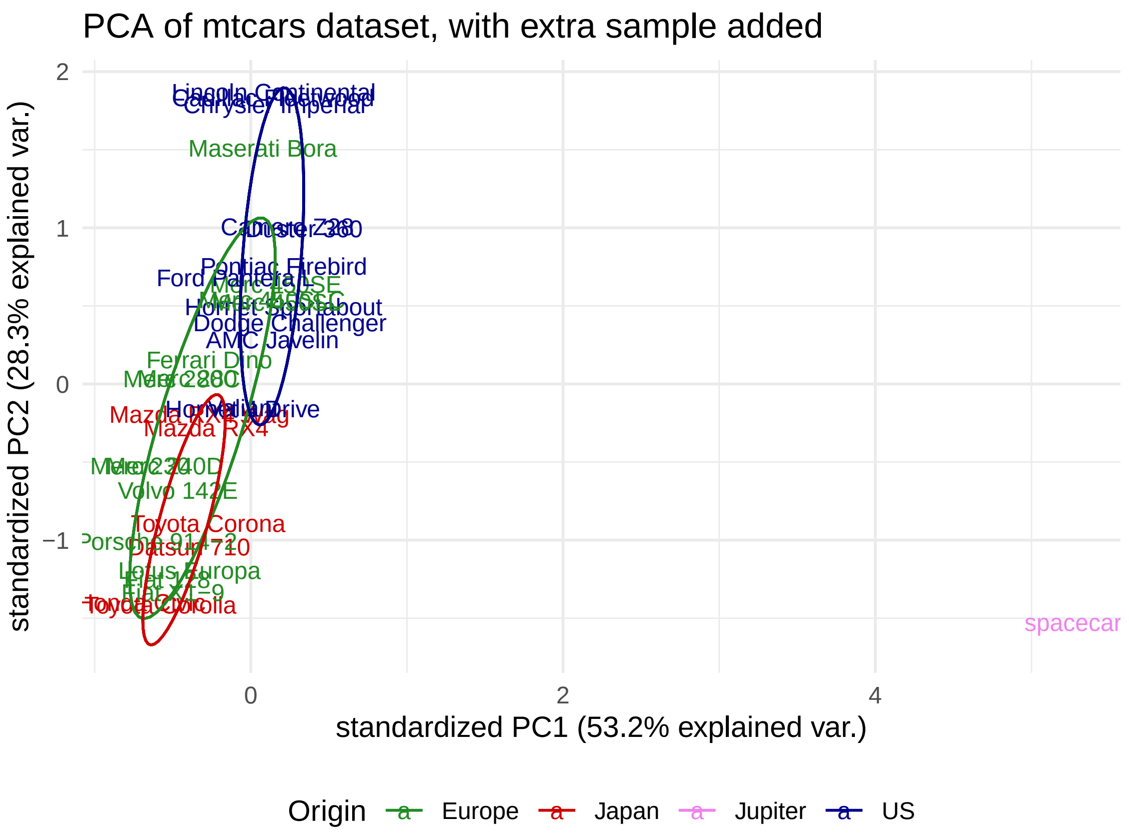
Can you add it to your existing dataset and see where it places in relation to the other cars?

You will add it to mtcars, creating mtcarsplus, then repeat your analysis. You might expect to be able to see which region's cars it is most like.

```
spacecar <- c(1000,60,50,500,0,0.5,2.5,0,1,0,0)

mtcarsplus <- rbind(mtcars, spacecar)

tail(mtcarsplus)
```

```
##                   mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L   15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino     19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora    15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E       21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
## 33             1000.0  60  50.0 500 0.00 0.500  2.5  0  1    0    0
```

```r
mtcars.countryplus <- c(mtcars_country, "Jupiter")

mtcarsplus.pca <- prcomp(mtcarsplus[,c(1:7,10,11)], center = T, scale = T)

ggbiplot(mtcarsplus.pca, ellipse = T, var.axes = F,
         labels=c(rownames(mtcars), "spacecar"), groups=mtcars.countryplus) +
  scale_colour_manual(name="Origin",
                      values= c("forest green", "red3", "violet", "dark blue"))+
  ggtitle("PCA of mtcars dataset, with extra sample added")+
  theme_minimal()+
  theme(legend.position = "bottom")
```



But that would be a naive assumption! The shape of the PCA has changed drastically, with the addition of this sample. When you consider this result in a bit more detail, it actually makes perfect sense. In the original dataset, you had strong correlations between certain variables (for example, cyl and mpg), which contributed to PC1, separating your groups from one another along this axis. However, when you perform the PCA with the extra sample, the same correlations are not present, which warps the whole dataset. In this case, the effect is particularly strong because your extra sample is an extreme outlier in multiple respects.

If you want to see how the new sample compares to the groups produced by the initial PCA, you need to project it onto that PCA.

PCA with factoextra

URL: http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/118-principal-component-analysis-in-r-prcomp-vs-princomp/

```
# install.packages("factoextra")
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
data(decathlon2)

str(decathlon2)
```

```
## 'data.frame':    27 obs. of  13 variables:
##  $ X100m       : num  11 10.8 11 11.3 11.1 ...
##  $ Long.jump   : num  7.58 7.4 7.23 7.09 7.3 7.31 6.81 7.56 6.97 7.27 ...
##  $ Shot.put    : num  14.8 14.3 14.2 15.2 13.5 ...
##  $ High.jump   : num  2.07 1.86 1.92 2.1 2.01 2.13 1.95 1.86 1.95 1.98 ...
##  $ X400m       : num  49.8 49.4 48.9 50.4 48.6 ...
##  $ X110m.hurdle: num  14.7 14.1 15 15.3 14.2 ...
##  $ Discus      : num  43.8 50.7 40.9 46.3 45.7 ...
##  $ Pole.vault  : num  5.02 4.92 5.32 4.72 4.42 4.42 4.92 4.82 4.72 4.62 ...
##  $ Javeline    : num  63.2 60.1 62.8 63.4 55.4 ...
##  $ X1500m      : num  292 302 280 276 268 ...
##  $ Rank        : int  1 2 4 5 7 8 9 10 11 12 ...
##  $ Points      : int  8217 8122 8067 8036 8004 7995 7802 7733 7708 7651 ...
##  $ Competition : Factor w/ 2 levels "Decastar","OlympicG": 1 1 1 1 1 1 1 1 1 1 ...
```

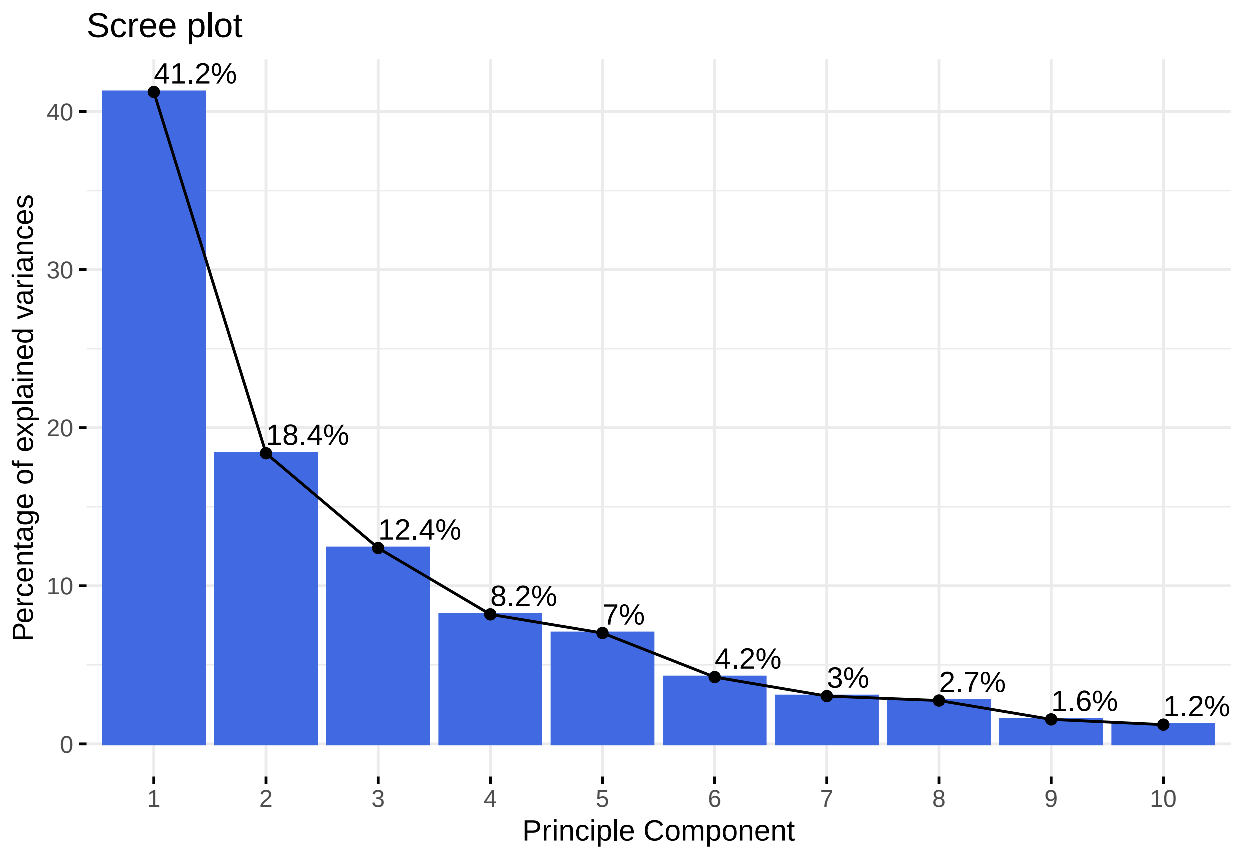
```
# View(decathlon2)

decathlon2.active <- decathlon2[1:23, 1:10]
head(decathlon2.active[, 1:5])
```

```
##           X100m Long.jump Shot.put High.jump X400m
## SEBRLE    11.04      7.58    14.83      2.07 49.81
## CLAY      10.76      7.40    14.26      1.86 49.37
## BERNARD   11.02      7.23    14.25      1.92 48.93
## YURKOV    11.34      7.09    15.19      2.10 50.42
## ZSIVOCZKY 11.13      7.30    13.48      2.01 48.62
## McMULLEN  10.83      7.31    13.76      2.13 49.91
```

```
res.pca <- prcomp(decathlon2.active, scale = TRUE)
summary(res.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.0308 1.3559 1.1132 0.90523 0.83759 0.65029 0.55007
## Proportion of Variance 0.4124 0.1839 0.1239 0.08194 0.07016 0.04229 0.03026
## Cumulative Proportion  0.4124 0.5963 0.7202 0.80213 0.87229 0.91458 0.94483
##                            PC8     PC9    PC10
## Standard deviation     0.52390 0.39398 0.3492
## Proportion of Variance 0.02745 0.01552 0.0122
## Cumulative Proportion  0.97228 0.98780 1.0000
```

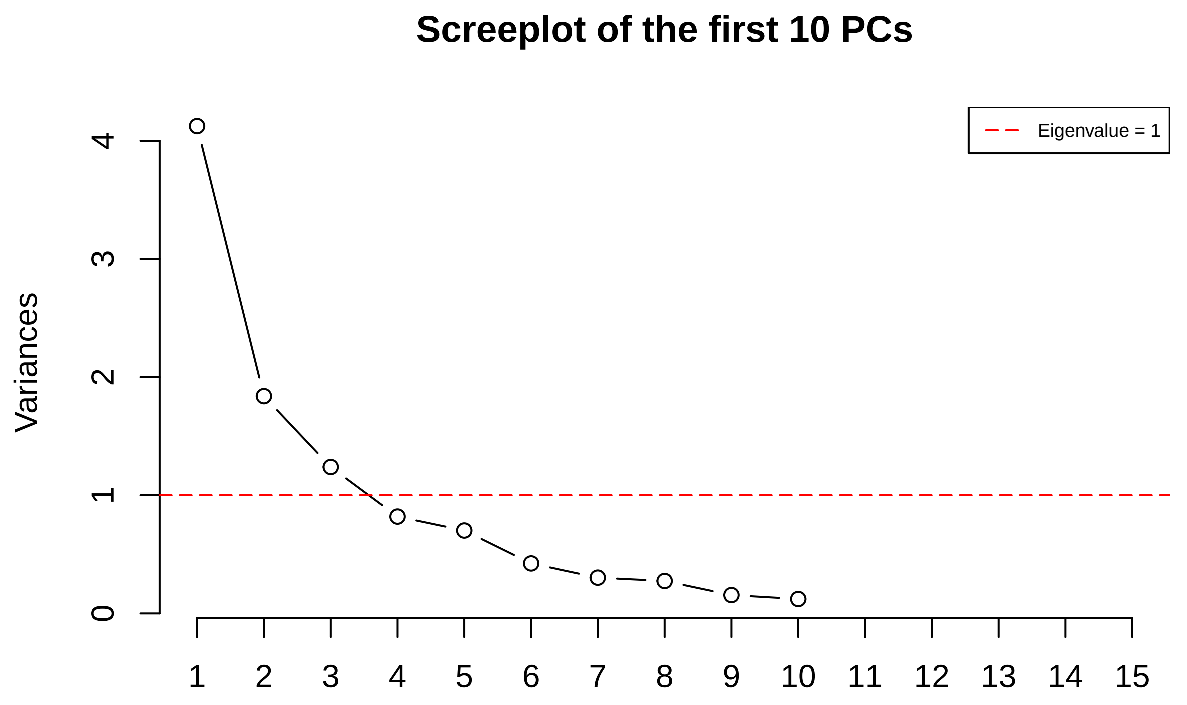
```r
fviz_eig(res.pca, xlab = "Principle Component", barfill = "royalblue",
         barcolor = "royalblue", linecolor = "black", addlabels = T)
```



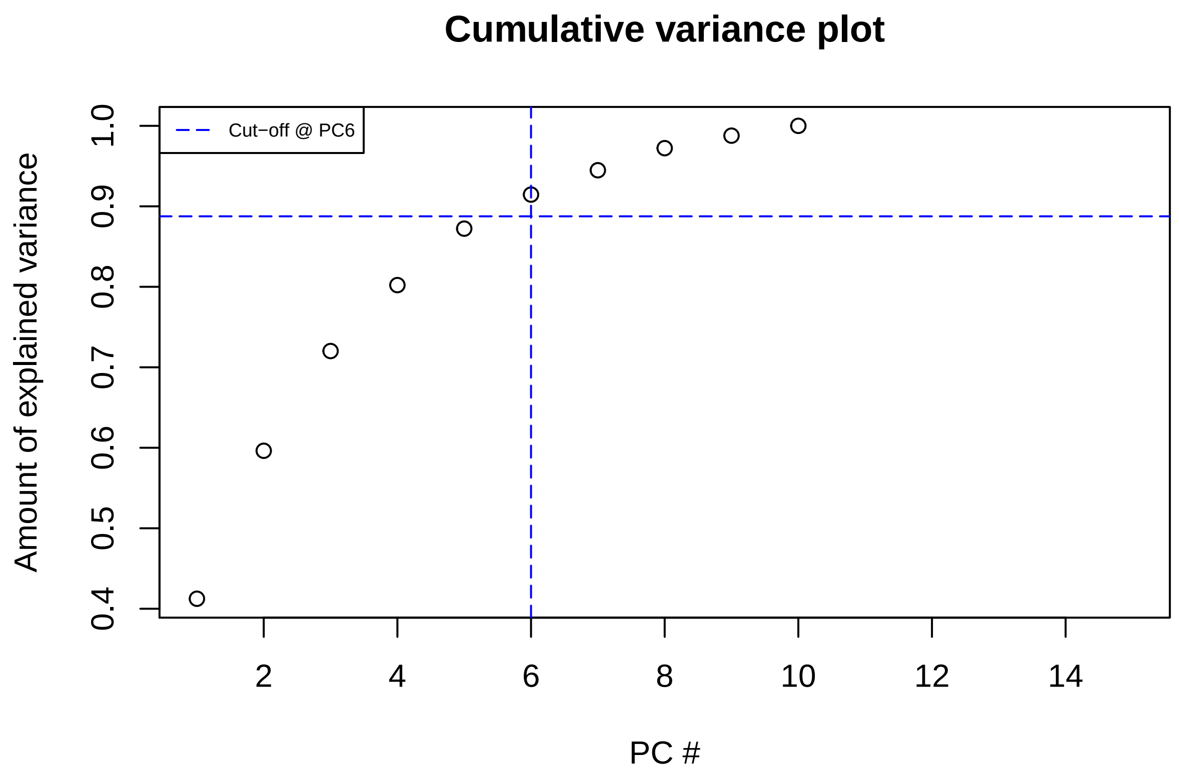
Right, so how many principle components do we want? We obviously want to be able to explain as much variance as possible but to do that we would need all 30 components, at the same time we want to reduce the number of dimensions so we definitely want less than 30!

Since we standardized our data and we now have the corresponding eigenvalues of each PC we can actually use these to draw a boundary for us. Since an eigenvalues <1 would mean that the component actually explains less than a single explanatory variable we would like to discard those. If our data is well suited for PCA we should be able to discard these components while retaining at least 70–80% of cumulative variance. Lets plot and see:

```r
# Scree Plot
screeplot(res.pca, type = "l", npcs = 15, main = "Screeplot of the first 10 PCs")
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),
       col=c("red"), lty=5, cex=0.6)
```



```
# Cumulative Plot
cumpro <- cumsum(res.pca$sdev^2 / sum(res.pca$sdev^2))
plot(cumpro[0:15], xlab = "PC #", ylab = "Amount of explained variance", main = "Cumulative variance pl
abline(v = 6, col="blue", lty=5)
abline(h = 0.88759, col="blue", lty=5)
legend("topleft", legend=c("Cut-off @ PC6"),
       col=c("blue"), lty=5, cex=0.6)
```

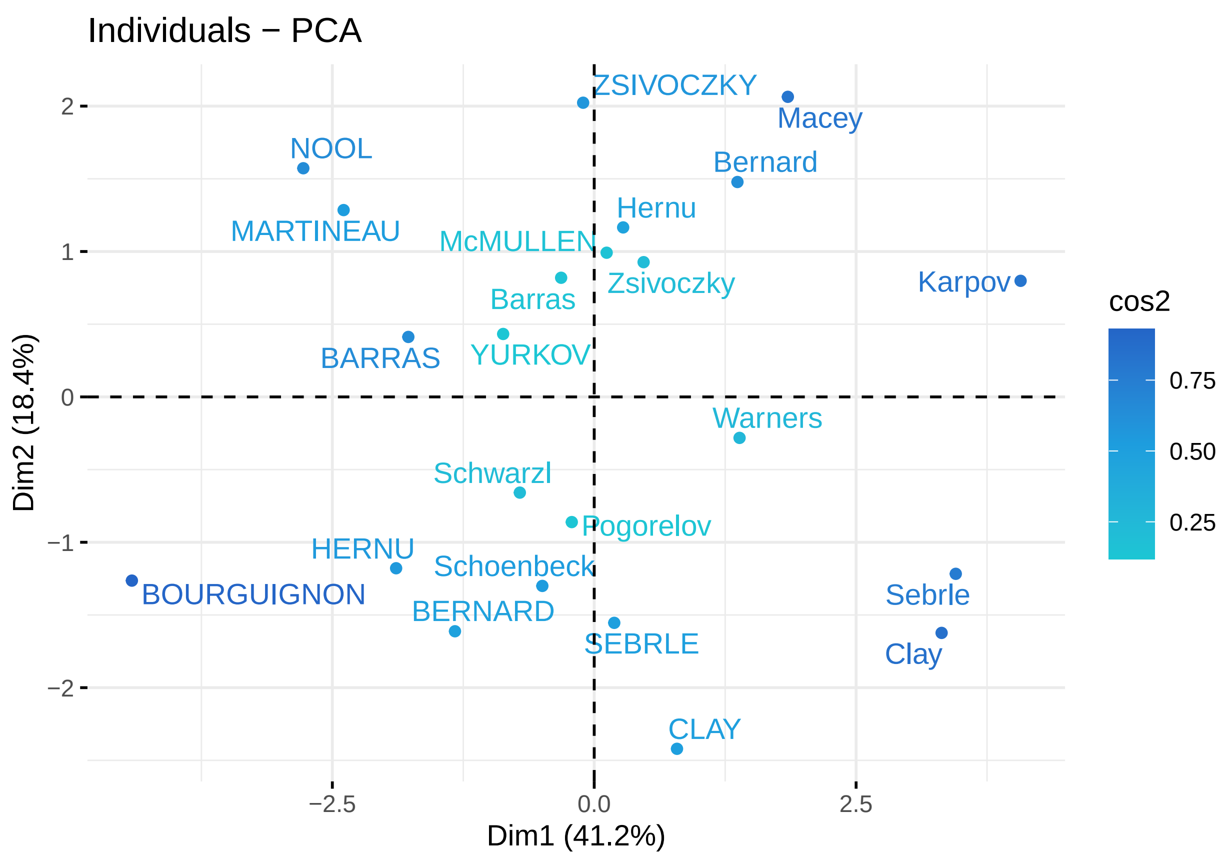


We notice that the first 3 components has an Eigenvalue >1 and explains almost 70% of variance, this is great! We can effectively reduce dimensionality from 10 to 3 while only "loosing" about 30% of variance!

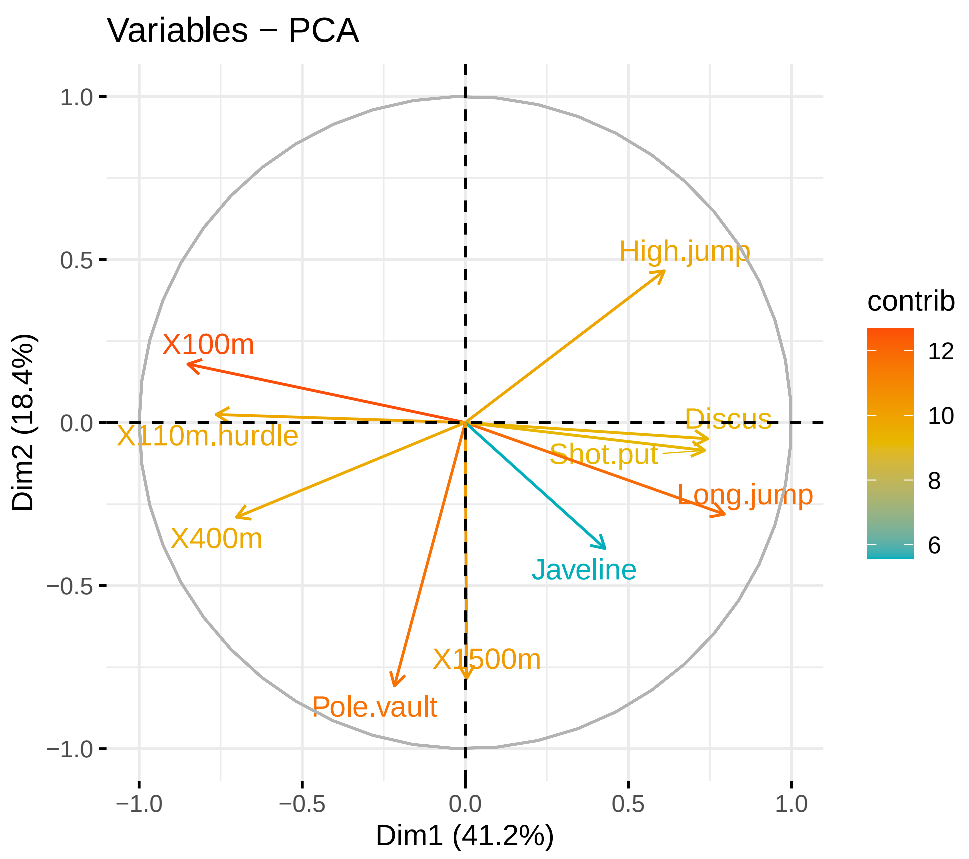
The selection of PC based on Eigenvalue >1 is known as Kaiser-Guttman Rule.

We also notice that we can actually explain more than 60% of variance with just the first two components.

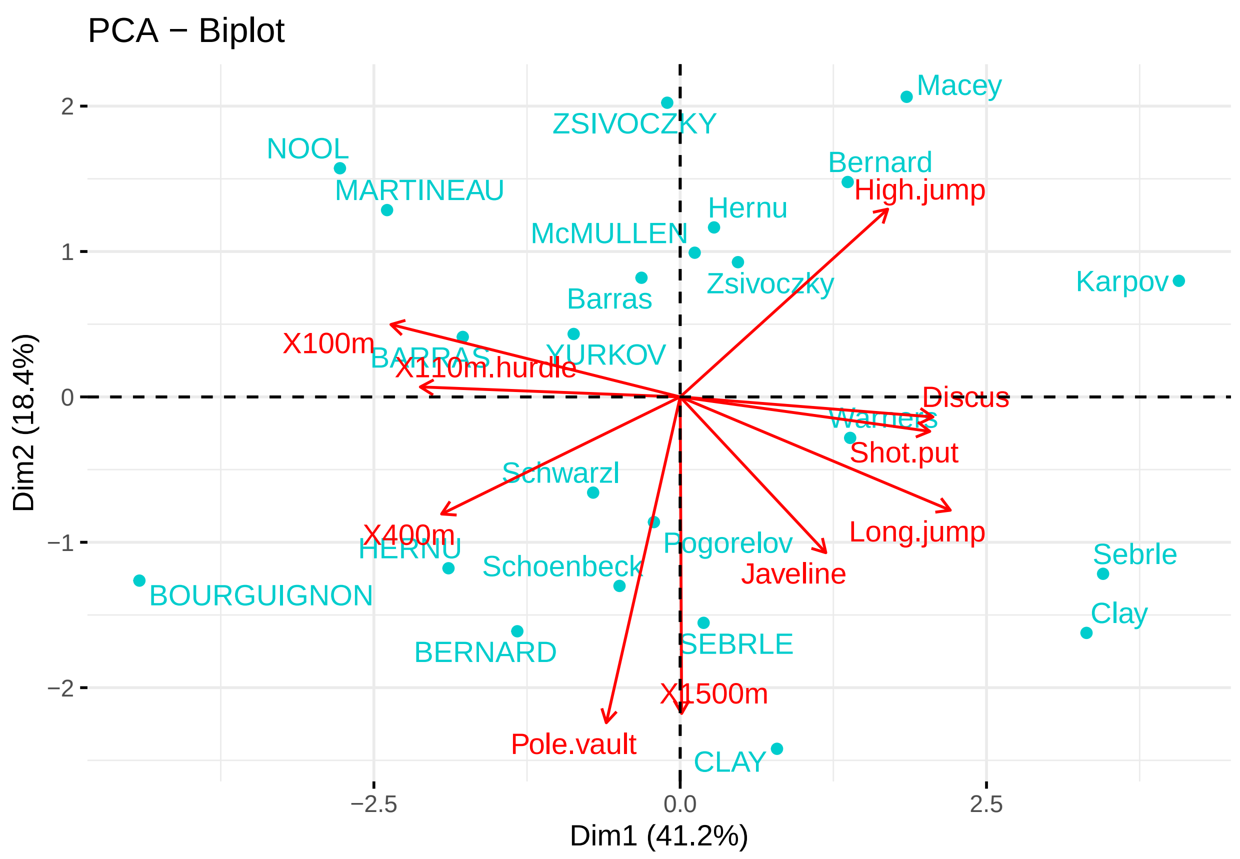
```
fviz_pca_ind(res.pca,
             col.ind = "cos2", # Color by the quality of representation
             repel = T, # Avoid text overlapping
             gradient.cols = c("#1CC6D4", "#1D9DDE","#2465C7"))
```



```r
fviz_pca_var(res.pca,
             col.var = "contrib",
             repel = T,
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```



```r
fviz_pca_biplot(res.pca,
                repel = T,
                col.ind = "cyan3",
                col.var = "red")
```



```
# Eigenvalues
eig.val <- get_eigenvalue(res.pca)
eig.val
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1    4.1242133        41.242133                    41.24213
## Dim.2    1.8385309        18.385309                    59.62744
## Dim.3    1.2391403        12.391403                    72.01885
## Dim.4    0.8194402         8.194402                    80.21325
## Dim.5    0.7015528         7.015528                    87.22878
## Dim.6    0.4228828         4.228828                    91.45760
## Dim.7    0.3025817         3.025817                    94.48342
## Dim.8    0.2744700         2.744700                    97.22812
## Dim.9    0.1552169         1.552169                    98.78029
## Dim.10   0.1219710         1.219710                   100.00000
```

```
# Results for Variables
res.var <- get_pca_var(res.pca)
res.var$coord          # Coordinates
```

```
##                     Dim.1       Dim.2       Dim.3       Dim.4      Dim.5
## X100m        -0.850625692  0.17939806 -0.30155643  0.03357320 -0.1944440
## Long.jump     0.794180641 -0.28085695  0.19054653 -0.11538956  0.2331567
## Shot.put      0.733912733 -0.08540412 -0.51759781  0.12846837 -0.2488129
## High.jump     0.610083985  0.46521415 -0.33008517  0.14455012  0.4027002
## X400m        -0.701603377 -0.29017826 -0.28353292  0.43082552  0.1039085
## X110m.hurdle -0.764125197  0.02474081 -0.44888733 -0.01689589  0.2242200
## Discus        0.743209016 -0.04966086 -0.17652518  0.39500915 -0.4082391
## Pole.vault   -0.217268042 -0.80745110 -0.09405773 -0.33898477 -0.2216853
## Javeline      0.428226639 -0.38610928 -0.60412432 -0.33173454  0.1978128
## X1500m        0.004278487 -0.78448019  0.21947068  0.44800961  0.2632527
##                     Dim.6        Dim.7        Dim.8       Dim.9       Dim.10
## X100m         0.035374780 -0.091336386 -0.104716925 -0.30306448  0.044417974
## Long.jump    -0.033727883 -0.154330810 -0.397380703 -0.05158951  0.029719453
## Shot.put     -0.239789034 -0.009886612  0.024359049  0.04778655  0.217451948
## High.jump    -0.284644846  0.028157465  0.084405578 -0.11213822 -0.133566774
## X400m        -0.049289996  0.286106008 -0.233552216  0.08216041 -0.034170673
## X110m.hurdle  0.002632395 -0.370072158 -0.008344682  0.16176025 -0.015629914
## Discus        0.198544870 -0.142725641 -0.039559255  0.01336209 -0.172590426
## Pole.vault   -0.327464549 -0.010393176  0.032914942 -0.02576874 -0.137211339
## Javeline      0.362097598  0.133564318  0.052841099 -0.04045397 -0.003854347
## X1500m        0.042050151 -0.111367083  0.194469730 -0.10224014  0.062834809
```

```r
res.var$contrib        # Contributions to the PCs
```

```
##                     Dim.1      Dim.2      Dim.3       Dim.4     Dim.5
## X100m        1.754429e+01  1.7505098  7.3386590  0.13755240  5.389252
## Long.jump    1.529317e+01  4.2904162  2.9300944  1.62485936  7.748815
## Shot.put     1.306014e+01  0.3967224 21.6204325  2.01407269  8.824401
## High.jump    9.024811e+00 11.7715838  8.7928883  2.54987951 23.115504
## X400m        1.193554e+01  4.5799296  6.4876363 22.65090599  1.539012
## X110m.hurdle 1.415754e+01  0.0332933 16.2612611  0.03483735  7.166193
## Discus       1.339309e+01  0.1341398  2.5147385 19.04132022 23.755756
## Pole.vault   1.144592e+00 35.4618611  0.7139512 14.02307063  7.005084
## Javeline     4.446377e+00  8.1086683 29.4531777 13.42963254  5.577615
## X1500m       4.438531e-04 33.4728757  3.8871610 24.49386930  9.878367
##                     Dim.6       Dim.7       Dim.8      Dim.9      Dim.10
## X100m         0.295915322  2.75705260  3.99520353 59.1740009  1.61756139
## Long.jump     0.269003613  7.87159392 57.53322220  1.7146826  0.72414393
## Shot.put     13.596858744  0.03230371  0.21618512  1.4712015 38.76768578
## High.jump    19.159607001  0.26202607  2.59565787  8.1015517 14.62649091
## X400m         0.574509906 27.05274658 19.87344405  4.3489667  0.95730504
## X110m.hurdle  0.001638634 45.26163460  0.02537025 16.8579392  0.20028870
## Discus        9.321746508  6.73226823  0.57016606  0.1150295 24.42174410
## Pole.vault   25.357622290  0.03569883  0.39472201  0.4278065 15.43559151
## Javeline     31.004964393  5.89573984  1.01729950  1.0543458  0.01217993
## X1500m        0.418133591  4.09893563 13.77872941  6.7344755  3.23700871
```

```r
res.var$cos2           # Quality of representation
```

```
##                     Dim.1        Dim.2       Dim.3        Dim.4      Dim.5
## X100m        7.235641e-01 0.0321836641 0.090936280 0.0011271597 0.03780845
## Long.jump    6.307229e-01 0.0788806285 0.036307981 0.0133147506 0.05436203
## Shot.put     5.386279e-01 0.0072938636 0.267907488 0.0165041211 0.06190783
## High.jump    3.722025e-01 0.2164242070 0.108956221 0.0208947375 0.16216747
## X400m        4.922473e-01 0.0842034209 0.080390914 0.1856106269 0.01079698
## X110m.hurdle 5.838873e-01 0.0006121077 0.201499837 0.0002854712 0.05027463
## Discus       5.523596e-01 0.0024662013 0.031161138 0.1560322304 0.16665918
## Pole.vault   4.720540e-02 0.6519772763 0.008846856 0.1149106765 0.04914437
## Javeline     1.833781e-01 0.1490803723 0.364966189 0.1100478063 0.03912992
## X1500m       1.830545e-05 0.6154091638 0.048167378 0.2007126089 0.06930197
##                     Dim.6        Dim.7        Dim.8        Dim.9       Dim.10
## X100m        1.251375e-03 0.0083423353 1.096563e-02 0.0918480768 1.972956e-03
## Long.jump    1.137570e-03 0.0238179990 1.579114e-01 0.0026614779 8.832459e-04
## Shot.put     5.749878e-02 0.0000977451 5.933633e-04 0.0022835540 4.728535e-02
## High.jump    8.102269e-02 0.0007928428 7.124302e-03 0.0125749811 1.784008e-02
## X400m        2.429504e-03 0.0818566479 5.454664e-02 0.0067503333 1.167635e-03
## X110m.hurdle 6.929502e-06 0.1369534023 6.963371e-05 0.0261663784 2.442942e-04
## Discus       3.942007e-02 0.0203706085 1.564935e-03 0.0001785453 2.978746e-02
## Pole.vault   1.072330e-01 0.0001080181 1.083393e-03 0.0006640282 1.882695e-02
## Javeline     1.311147e-01 0.0178394271 2.792182e-03 0.0016365234 1.485599e-05
## X1500m       1.768215e-03 0.0124026272 3.781848e-02 0.0104530472 3.948213e-03
```

```
# Results for individuals
res.ind <- get_pca_ind(res.pca)
res.ind$coord          # Coordinates
```

```
##                  Dim.1      Dim.2       Dim.3       Dim.4         Dim.5
## SEBRLE       0.1912074 -1.5541282 -0.62836882  0.08205241  1.1426139415
## CLAY         0.7901217 -2.4204156  1.35688701  1.26984296 -0.8068483724
## BERNARD     -1.3292592 -1.6118687 -0.19614996 -1.92092203  0.0823428202
## YURKOV      -0.8694134  0.4328779 -2.47398223  0.69723814  0.3988584116
## ZSIVOCZKY   -0.1057450  2.0233632  1.30493117 -0.09929630 -0.1970241089
## McMULLEN     0.1185550  0.9916237  0.84355824  1.31215266  1.5858708644
## MARTINEAU   -2.3923532  1.2849234 -0.89816842  0.37309771 -2.2433515889
## HERNU       -1.8910497 -1.1784614 -0.15641037  0.89130068 -0.1267412520
## BARRAS      -1.7744575  0.4125321  0.65817750  0.22872866 -0.2338366980
## NOOL        -2.7770058  1.5726757  0.60724821 -1.55548081  1.4241839810
## BOURGUIGNON -4.4137335 -1.2635770 -0.01003734  0.66675478  0.4191518468
## Sebrle       3.4514485 -1.2169193 -1.67816711 -0.80870696 -0.0250530746
## Clay         3.3162243 -1.6232908 -0.61840443 -0.31679906  0.5691645854
## Karpov       4.0703560  0.7983510  1.01501662  0.31336354 -0.7974259553
## Macey        1.8484623  2.0638828 -0.97928455  0.58469073 -0.0002157834
## Warners      1.3873514 -0.2819083  1.99969621 -1.01959817 -0.0405401497
## Zsivoczky    0.4715533  0.9267436 -1.72815525 -0.18483138  0.4073029909
## Hernu        0.2763118  1.1657260  0.17056375 -0.84869401 -0.6894795441
## Bernard      1.3672590  1.4780354  0.83137913  0.74531557  0.8598016482
## Schwarzl    -0.7102777 -0.6584251  1.04075176 -0.92717510 -0.2887568007
## Pogorelov   -0.2143524 -0.8610557  0.29761010  1.35560294 -0.0150531057
## Schoenbeck  -0.4953166 -1.3000530  0.10300360 -0.24927712 -0.6452257128
## Barras      -0.3158867  0.8193681 -0.86169481 -0.58935985 -0.7797389436
##                   Dim.6       Dim.7        Dim.8        Dim.9      Dim.10
## SEBRLE      -0.46389755 -0.20796012  0.043460568 -0.659344137  0.03273238
## CLAY         1.30420016 -0.21291866  0.617240611 -0.060125359 -0.31716015
## BERNARD     -0.40062867 -0.40643754  0.703856040  0.170083313 -0.09908142
## YURKOV       0.10286344 -0.32487448  0.114996135 -0.109524039 -0.11969720
## ZSIVOCZKY    0.89554111  0.08825624 -0.202341299 -0.523103099 -0.34842265
## McMULLEN     0.18657283  0.47828432  0.293089967 -0.105623196 -0.39317797
## MARTINEAU   -0.45666350 -0.29975522 -0.291628488 -0.223417655 -0.61640509
## HERNU        0.43623496 -0.56609980 -1.529404317  0.006184409  0.55368016
## BARRAS       0.09026010  0.21594095  0.682583078 -0.669282042  0.53085420
## NOOL         0.49716399 -0.53205687 -0.433385655 -0.115777808 -0.09622142
## BOURGUIGNON -0.08200220 -0.59833739  0.563619921  0.525814030  0.05855882
## Sebrle      -0.08279306  0.01016177 -0.030585843 -0.847210682  0.21970353
## Clay         0.77715960  0.25750851 -0.580638301  0.409776590 -0.61601933
## Karpov      -0.32958134 -1.36365568  0.345306381  0.193055107  0.21721852
## Macey       -0.19728082 -0.26927772 -0.363219506  0.368260269  0.21249474
## Warners     -0.55673300 -0.26739400 -0.109470797  0.180283071  0.24208420
## Zsivoczky   -0.11383190  0.03991159  0.538039776  0.585966156 -0.14271715
## Hernu       -0.33168404  0.44308686  0.247293566  0.066908586 -0.20868256
## Bernard     -0.32806564  0.36357920  0.006165316  0.279488675  0.32067773
## Schwarzl    -0.68891640  0.56568604 -0.687053339 -0.008358849 -0.30211546
## Pogorelov   -1.59379599  0.78370119 -0.037623661 -0.130531397 -0.03697576
## Schoenbeck   0.16172381  0.85752368 -0.255850722  0.564222295  0.29680481
## Barras       1.17415412  0.94512710  0.365550568  0.102255763  0.61186706
```

```r
res.ind$contrib         # Contributions to the PCs
```

```
##                   Dim.1      Dim.2        Dim.3       Dim.4        Dim.5
## SEBRLE       0.03854254  5.7118249 1.385418e+00  0.03572215 8.091161e+00
## CLAY         0.65814114 13.8541889 6.460097e+00  8.55568792 4.034555e+00
## BERNARD      1.86273218  6.1441319 1.349983e-01 19.57827284 4.202070e-02
## YURKOV       0.79686310  0.4431309 2.147558e+01  2.57939100 9.859373e-01
## ZSIVOCZKY    0.01178829  9.6816398 5.974848e+00  0.05231437 2.405750e-01
## McMULLEN     0.01481737  2.3253860 2.496789e+00  9.13531719 1.558646e+01
## MARTINEAU    6.03367104  3.9044125 2.830527e+00  0.73858431 3.118936e+01
## HERNU        3.76996156  3.2842176 8.583863e-02  4.21505626 9.955149e-02
## BARRAS       3.31942012  0.4024544 1.519980e+00  0.27758505 3.388731e-01
## NOOL         8.12988880  5.8489726 1.293851e+00 12.83761115 1.257025e+01
## BOURGUIGNON 20.53729577  3.7757623 3.534995e-04  2.35877858 1.088816e+00
## Sebrle      12.55838616  3.5020697 9.881482e+00  3.47006223 3.889859e-03
## Clay        11.59361384  6.2315181 1.341828e+00  0.53250375 2.007648e+00
## Karpov      17.46609555  1.5072627 3.614914e+00  0.52101693 3.940874e+00
## Macey        3.60207087 10.0732890 3.364879e+00  1.81387486 2.885677e-07
## Warners      2.02910262  0.1879390 1.403071e+01  5.51585696 1.018550e-02
## Zsivoczky    0.23441891  2.0310492 1.047894e+01  0.18126182 1.028128e+00
## Hernu        0.08048777  3.2136178 1.020764e-01  3.82170515 2.946148e+00
## Bernard      1.97075488  5.1661961 2.425213e+00  2.94737426 4.581507e+00
## Schwarzl     0.53184785  1.0252129 3.800546e+00  4.56119277 5.167449e-01
## Pogorelov    0.04843819  1.7533304 3.107757e-01  9.75034337 1.404313e-03
## Schoenbeck   0.25864068  3.9969003 3.722687e-02  0.32970059 2.580092e+00
## Barras       0.10519467  1.5876667 2.605305e+00  1.84296038 3.767994e+00
##                   Dim.6       Dim.7        Dim.8        Dim.9      Dim.10
## SEBRLE       2.21256620 0.621426384 2.992045e-02 12.177477305  0.03819185
## CLAY        17.48801877 0.651413899 6.035125e+00  0.101262442  3.58568943
## BERNARD      1.65019840 2.373652810 7.847747e+00  0.810319793  0.34994507
## YURKOV       0.10878629 1.516564073 2.094806e-01  0.336009790  0.51072064
## ZSIVOCZKY    8.24561722 0.111923276 6.485544e-01  7.664919832  4.32741147
## McMULLEN     0.35788945 3.287016354 1.360753e+00  0.312501167  5.51053518
## MARTINEAU    2.14409841 1.291109482 1.347216e+00  1.398195851 13.54402896
## HERNU        1.95655942 4.604850849 3.705288e+01  0.001071345 10.92781554
## BARRAS       0.08376135 0.670038259 7.380544e+00 12.547331617 10.04537028
## NOOL         2.54127369 4.067669683 2.975270e+00  0.375477289  0.33003418
## BOURGUIGNON  0.06913582 5.144247534 5.032108e+00  7.744571086  0.12223626
## Sebrle       0.07047579 0.001483775 1.481898e-02 20.105546253  1.72063803
## Clay         6.20972751 0.952824148 5.340583e+00  4.703566841 13.52708188
## Karpov       1.11680500 26.720158115 1.888802e+00  1.043988269  1.68193477
## Macey        0.40014909  1.041910483 2.089853e+00  3.798767930  1.60957713
## Warners      3.18673563  1.027384225 1.898339e-01  0.910422384  2.08904756
## Zsivoczky    0.13322327  0.022889042 4.585705e+00  9.617852173  0.72605208
## Hernu        1.13110069  2.821027418 9.687304e-01  0.125399768  1.55234328
## Bernard      1.10655655  1.899449022 6.021268e-04  2.188071254  3.66566729
## Schwarzl     4.87961053  4.598122119 7.477531e+00  0.001957159  3.25357879
## Pogorelov   26.11665608  8.825322559 2.242329e-02  0.477268755  0.04873597
## Schoenbeck   0.26890572 10.566272800 1.036933e+00  8.917302863  3.14020004
## Barras      14.17432302 12.835417603 2.116763e+00  0.292892746 13.34533825
```

```r
res.ind$cos2           # Quality of representation
```

```
##                   Dim.1      Dim.2        Dim.3       Dim.4        Dim.5
## SEBRLE      0.007530179 0.49747323 8.132523e-02 0.001386688 2.689027e-01
## CLAY        0.048701249 0.45701660 1.436281e-01 0.125791741 5.078506e-02
## BERNARD     0.197199804 0.28996555 4.294015e-03 0.411819183 7.567259e-04
## YURKOV      0.096109800 0.02382571 7.782303e-01 0.061812637 2.022798e-02
## ZSIVOCZKY   0.001574385 0.57641944 2.397542e-01 0.001388216 5.465497e-03
## McMULLEN    0.002175437 0.15219499 1.101379e-01 0.266486530 3.892621e-01
## MARTINEAU   0.404013915 0.11654676 5.694575e-02 0.009826320 3.552552e-01
## HERNU       0.399282749 0.15506199 2.731529e-03 0.088699901 1.793538e-03
## BARRAS      0.616241975 0.03330700 8.478249e-02 0.010239088 1.070152e-02
## NOOL        0.489872515 0.15711146 2.342405e-02 0.153694675 1.288433e-01
## BOURGUIGNON 0.859698130 0.07045912 4.446015e-06 0.019618511 7.753120e-03
## Sebrle      0.675380606 0.08395940 1.596674e-01 0.037079012 3.558507e-05
## Clay        0.687592867 0.16475409 2.391051e-02 0.006274965 2.025440e-02
## Karpov      0.783666922 0.03014772 4.873187e-02 0.004644764 3.007790e-02
## Macey       0.363436037 0.45308203 1.020057e-01 0.036362957 4.952707e-09
## Warners     0.255651956 0.01055582 5.311341e-01 0.138081100 2.182965e-04
## Zsivoczky   0.045053176 0.17401353 6.051030e-01 0.006921739 3.361236e-02
## Hernu       0.024824321 0.44184663 9.459148e-03 0.234196727 1.545686e-01
## Bernard     0.289347476 0.33813318 1.069834e-01 0.085980212 1.144234e-01
## Schwarzl    0.116721435 0.10030142 2.506043e-01 0.198892209 1.929118e-02
## Pogorelov   0.007803472 0.12591966 1.504272e-02 0.312101619 3.848427e-05
## Schoenbeck  0.067070098 0.46204603 2.900467e-03 0.016987442 1.138116e-01
## Barras      0.018972684 0.12765099 1.411800e-01 0.066043061 1.156018e-01
##                    Dim.6        Dim.7        Dim.8        Dim.9       Dim.10
## SEBRLE      0.0443241299 8.907507e-03 3.890334e-04 8.954067e-02 0.0002206741
## CLAY        0.1326907339 3.536548e-03 2.972084e-02 2.820119e-04 0.0078471026
## BERNARD     0.0179131165 1.843634e-02 5.529104e-02 3.228572e-03 0.0010956493
## YURKOV      0.0013453555 1.341980e-02 1.681440e-03 1.525225e-03 0.0018217256
## ZSIVOCZKY   0.1129176906 1.096685e-03 5.764478e-03 3.852703e-02 0.0170924251
## McMULLEN    0.0053876990 3.540616e-02 1.329562e-02 1.726733e-03 0.0239268142
## MARTINEAU   0.0147210347 6.342774e-03 6.003515e-03 3.523552e-03 0.0268211980
## HERNU       0.0212478795 3.578167e-02 2.611676e-01 4.270425e-06 0.0342288717
## BARRAS      0.0015944528 9.126203e-03 9.118662e-02 8.766746e-02 0.0551531863
## NOOL        0.0157010551 1.798232e-02 1.193105e-02 8.514912e-04 0.0005881295
## BOURGUIGNON 0.0002967459 1.579887e-02 1.401866e-02 1.220108e-02 0.0001513277
## Sebrle      0.0003886276 5.854423e-06 5.303795e-05 4.069384e-02 0.0027366539
## Clay        0.0377627839 4.145976e-03 2.107924e-02 1.049876e-02 0.0237264222
## Karpov      0.0051379747 8.795817e-02 5.639959e-03 1.762907e-03 0.0022318265
## Macey       0.0041397727 7.712721e-03 1.403282e-02 1.442502e-02 0.0048028954
## Warners     0.0411689767 9.496848e-03 1.591742e-03 4.317040e-03 0.0077841113
## Zsivoczky   0.0026253777 3.227467e-04 5.865332e-02 6.956790e-02 0.0041268259
## Hernu       0.0357707217 6.383462e-02 1.988402e-02 1.455601e-03 0.0141595965
## Bernard     0.0166586433 2.046050e-02 5.883405e-06 1.209056e-02 0.0159167991
## Schwarzl    0.1098063093 7.403638e-02 1.092132e-01 1.616543e-05 0.0211173850
## Pogorelov   0.4314162233 1.043115e-01 2.404103e-04 2.893750e-03 0.0002322016
## Schoenbeck  0.0071500829 2.010275e-01 1.789520e-02 8.702893e-02 0.0240826922
## Barras      0.2621297474 1.698426e-01 2.540745e-02 1.988116e-03 0.0711836486
```

Testing on the remaining data

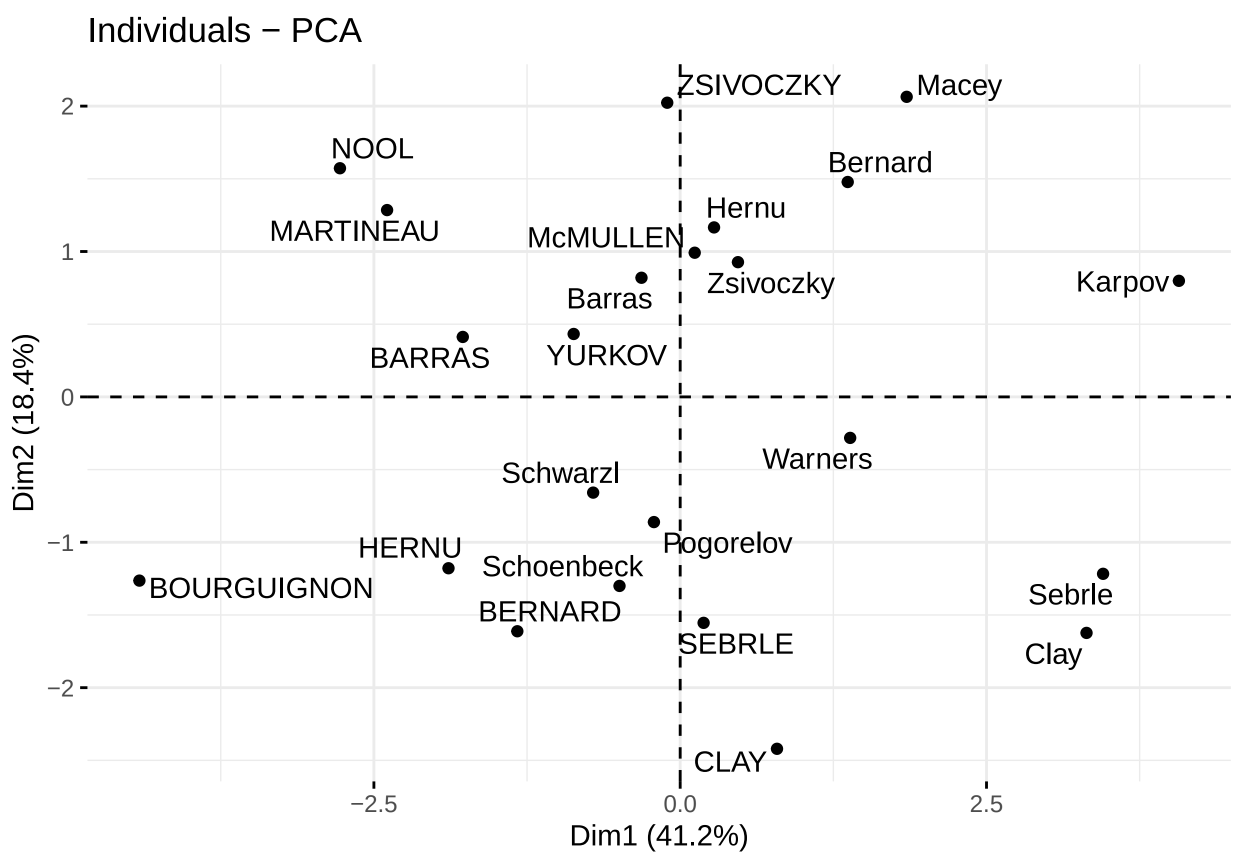
```
# Data for the supplementary individuals
ind.sup <- decathlon2[24:27, 1:10]
ind.sup[, 1:5]
```

```
##         X100m Long.jump Shot.put High.jump X400m
## KARPOV  11.02      7.30    14.77      2.04 48.37
## WARNERS 11.11      7.60    14.31      1.98 48.68
## Nool    10.80      7.53    14.26      1.88 48.81
## Drews   10.87      7.38    13.07      1.88 48.51
```

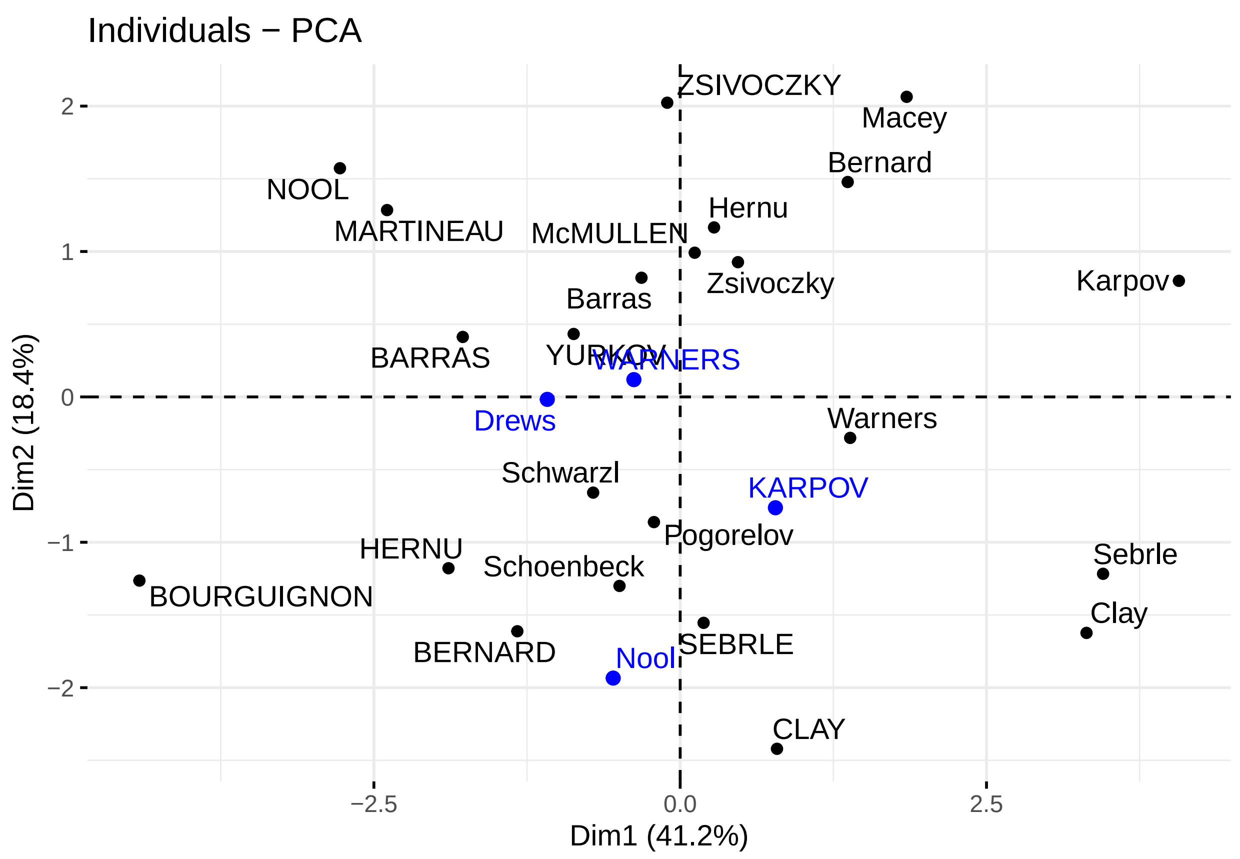
```
ind.sup.coord <- predict(res.pca, newdata = ind.sup)
ind.sup.coord[, 1:4]
```

```
##                PC1         PC2       PC3        PC4
## KARPOV   0.7772521 -0.76237804 1.5971253  1.6863286
## WARNERS -0.3779697  0.11891968 1.7005146 -0.6908084
## Nool    -0.5468405 -1.93402211 0.4724184 -2.2283706
## Drews   -1.0848227 -0.01703198 2.9818031 -1.5006207
```

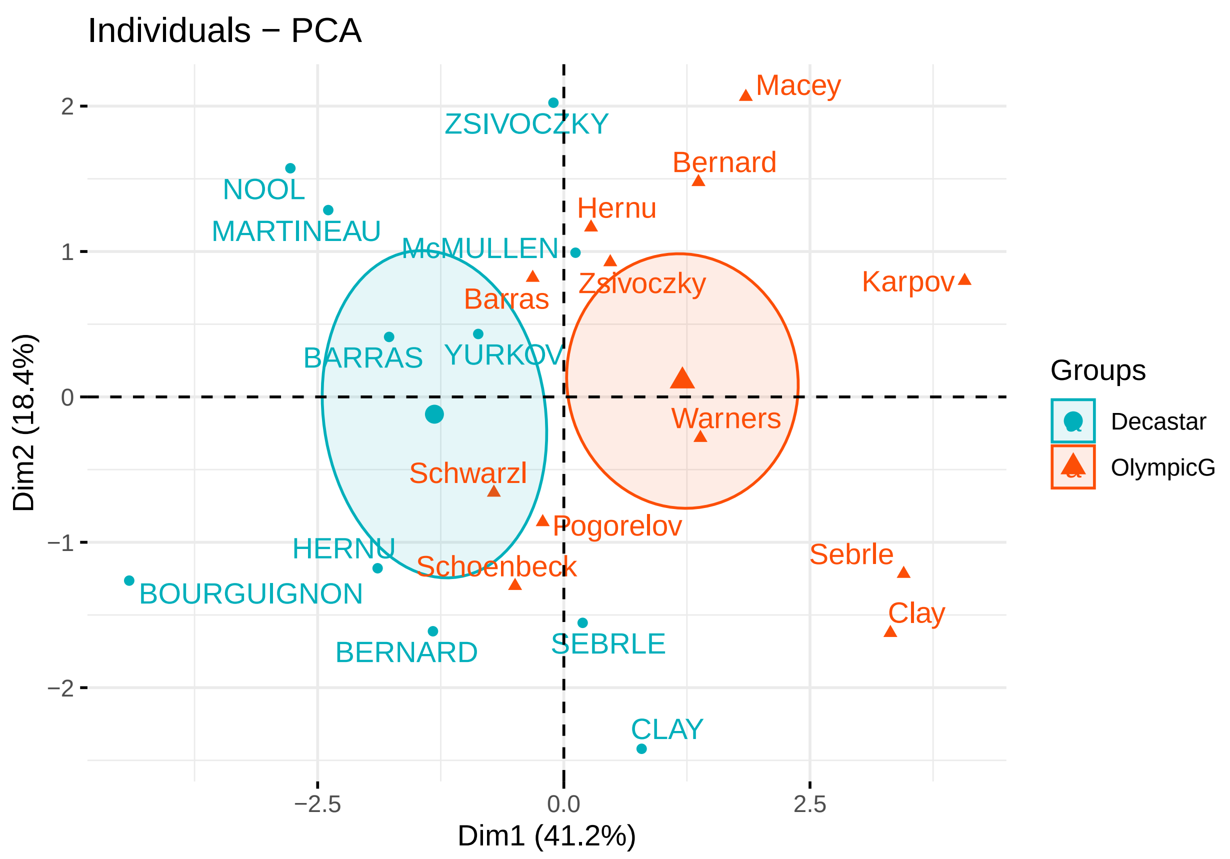
```
# Plot of active individuals
p <- fviz_pca_ind(res.pca, repel = TRUE)
p
```



```r
# Add supplementary individuals
fviz_add(p, ind.sup.coord, color ="blue", repel = T)
```



```r
groups <- as.factor(decathlon2$Competition[1:23])
fviz_pca_ind(res.pca,
             col.ind = groups, # color by groups
             palette = c("#00AFBB",  "#FC4E07"),
             addEllipses = T, # Concentration ellipses
             ellipse.type = "confidence",
             legend.title = "Groups",
             repel = T)
```



PCA from Towardsdatascience

URL: https://towardsdatascience.com/principal-component-analysis-pca-101-using-r-361f4c53a9ff

Tumor classification For this article we'll be using the Breast Cancer Wisconsin data set from the UCI Machine learning repo as our data.

```
wdbc <- read.csv(url('http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/w

features <- c("radius", "texture", "perimeter", "area",
              "smoothness", "compactness", "concavity",
              "concave_points", "symmetry", "fractal_dimension")

names(wdbc) <- c("id", "diagnosis",
                 paste0(features,"_mean"),
                 paste0(features,"_se"),
                 paste0(features,"_worst"))
```

Why PCA? Right, so now we've loaded our data and find ourselves with 30 variables (thus excluding our response "diagnosis" and the irrelevant ID-variable). Now some of you might be saying "30 variable is a lot" and some might say "Pfft.. Only 30? I've worked with THOUSANDS!!" but rest assured that this is equally applicable in either scenario..!

There's a few pretty good reasons to use PCA. The plot at the very beginning af the article is a great example of how one would plot multi-dimensional data by using PCA, we actually capture 63.3% (Dim1 44.3% + Dim2 19%) of variance in the entire dataset by just using those two principal components, pretty good when taking into consideration that the original data consisted of 30 features which would be impossible to plot in any meaningful way.

A very powerful consideration is to acknowledge that we never specified a response variable or anything else in our PCA-plot indicating whether a tumor was "benign" or "malignant". It simply turns out that when we try to describe variance in the data using the linear combinations of the PCA we find some pretty obvious clustering and separation between the "benign" and "malignant" tumors! This makes a great case for developing a classification model based on our features!

Another major "feature" (no pun intended) of PCA is that it can actually directly improve performance of your models.

PCA on our tumor data So now we understand a bit about how PCA works and that should be enough for now. Lets actually try it out:

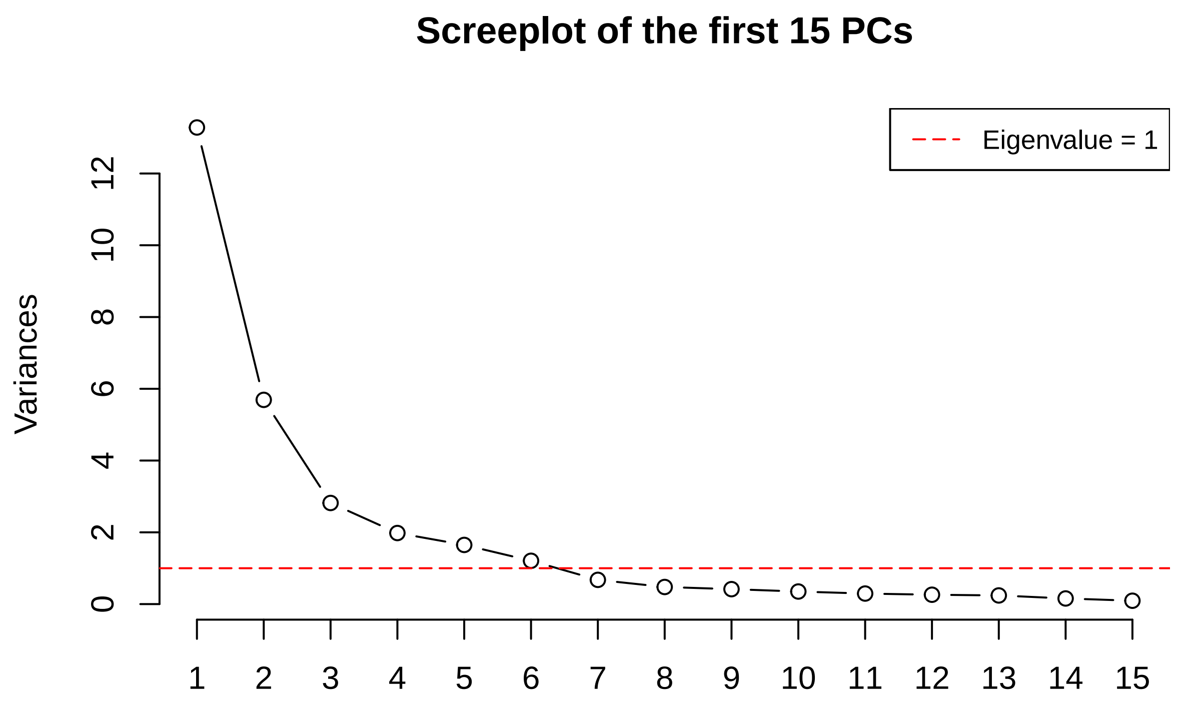
```
wdbc.pr <- prcomp(wdbc[c(3:32)], center = TRUE, scale = TRUE)
summary(wdbc.pr)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                            PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

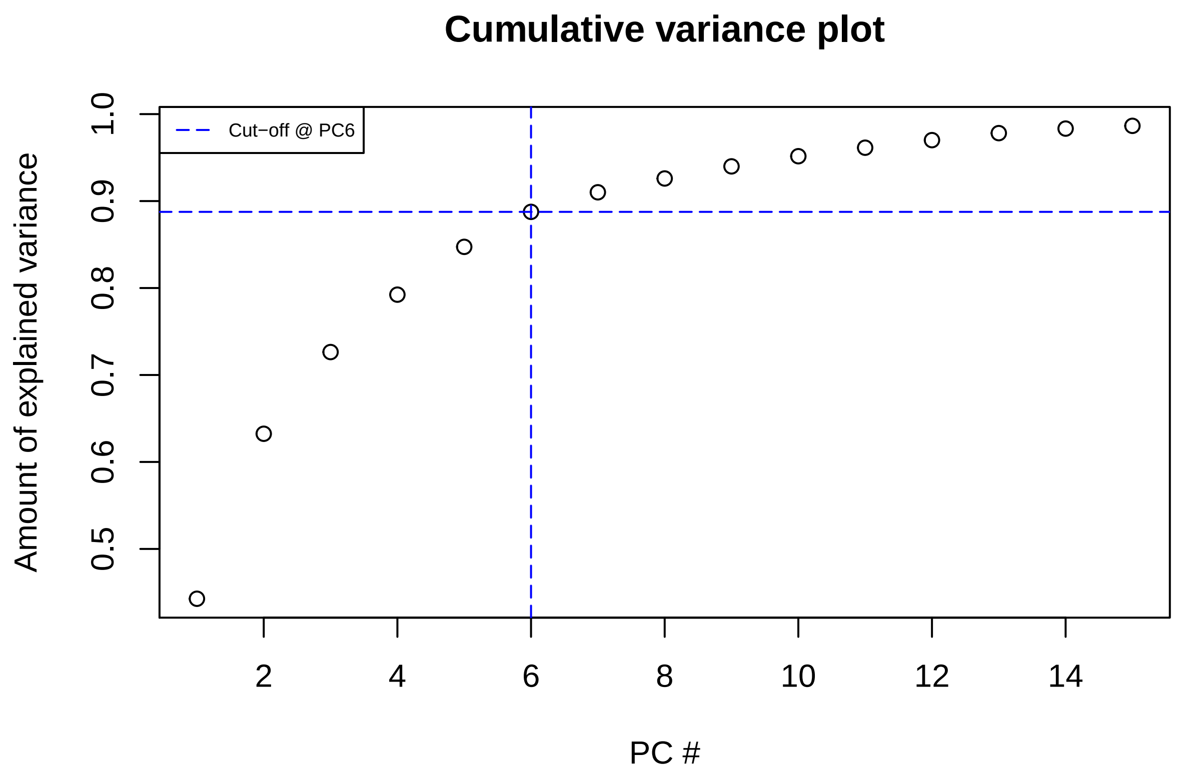
So, how many components do we want? We obviously want to be able to explain as much variance as possible but to do that we would need all 30 components, at the same time we want to reduce the number of dimensions so we definitely want less than 30!

Since we standardized our data and we now have the corresponding eigenvalues of each PC we can actually use these to draw a boundary for us. Since an eigenvalues <1 would mean that the component actually explains less than a single explanatory variable we would like to discard those. If our data is well suited for PCA we should be able to discard these components while retaining at least 70–80% of cumulative variance. Lets plot and see:

```r
screeplot(wdbc.pr,
          type = "l",
          npcs = 15,
          main = "Screeplot of the first 15 PCs")
abline(h = 1, col = "red", lty = 5)
legend("topright", legend = c("Eigenvalue = 1"), col = "red",
       lty = 5, cex = 0.8)
```



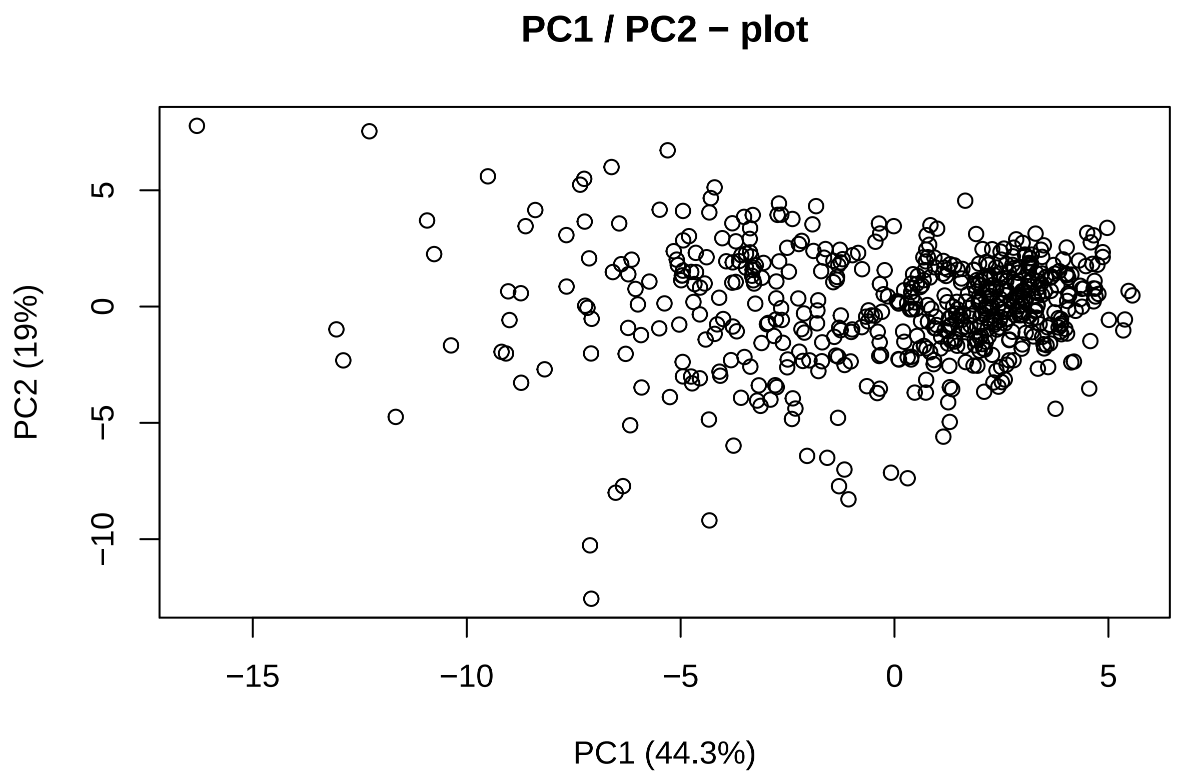
```r
cumpro <- cumsum(wdbc.pr$sdev ^ 2 / sum(wdbc.pr$sdev ^ 2))
plot(cumpro[0:15],
     xlab = "PC #",
     ylab = "Amount of explained variance",
     main = "Cumulative variance plot")

abline(v = 6, col = "blue", lty = 5)
abline(h = 0.88759, col = "blue", lty = 5)
legend( "topleft", legend = c("Cut-off @ PC6"),
        col = c("blue"),lty = 5, cex = 0.6)
```



We notice is that the first 6 components has an Eigenvalue >1 and explains almost 90% of variance, this is great! We can effectively reduce dimensionality from 30 to 6 while only "loosing" about 10% of variance!

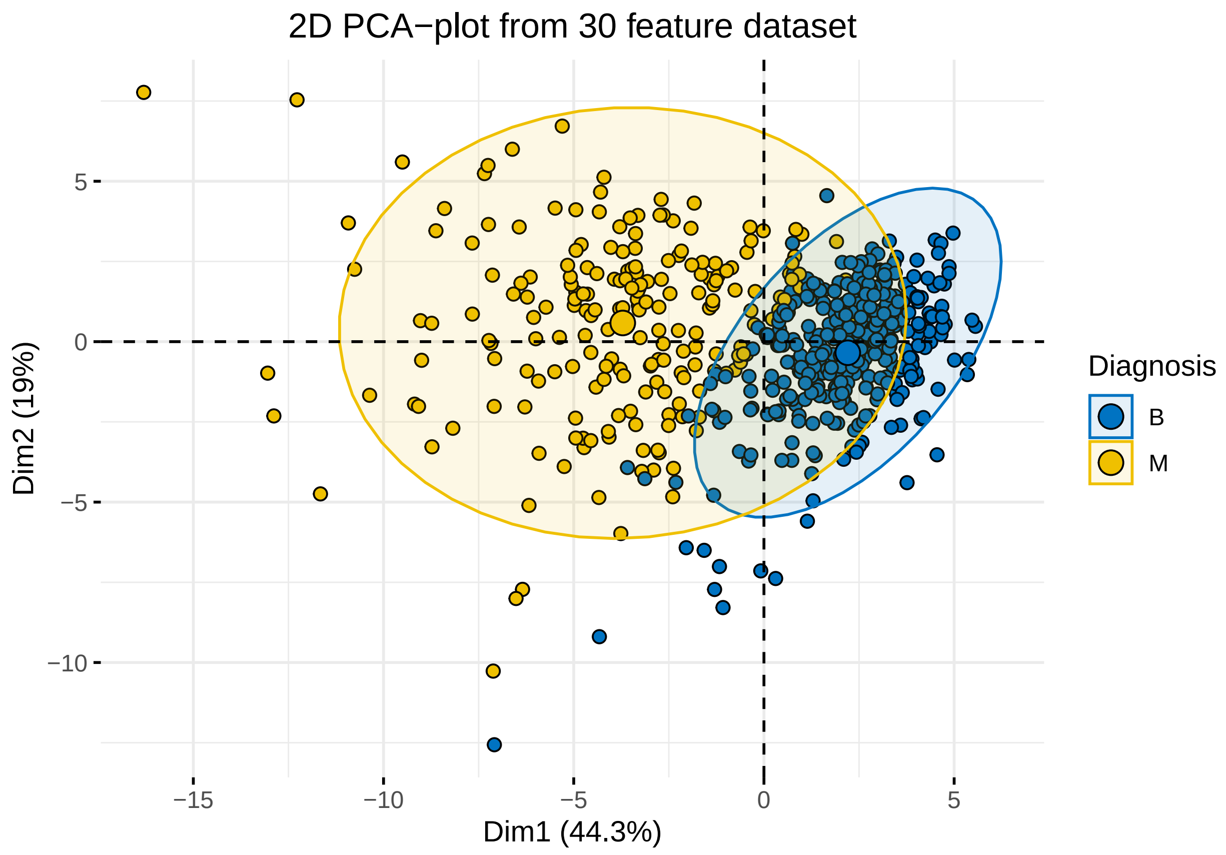
We also notice that we can actually explain more than 60% of variance with just the first two components. Let's try plotting these:

```
plot(wdbc.pr$x[,1],wdbc.pr$x[,2], xlab="PC1 (44.3%)", ylab = "PC2 (19%)", main = "PC1 / PC2 - plot")
```



Alright, this isn't really too telling but consider for a moment that this is representing 60%+ of variance in a 30 dimensional dataset. But what do we see from this? There's some clustering going on in the upper/middle-right. Lets also consider for a moment what the goal of this analysis actually is. We want to explain difference between malignant and benign tumors. Let's actually add the response variable (diagnosis) to the plot and see if we can make better sense of it:

```r
library("factoextra")
fviz_pca_ind(wdbc.pr, geom.ind = "point", pointshape = 21,
             pointsize = 2, fill.ind = wdbc$diagnosis,
             col.ind = "black", palette = "jco", addEllipses = TRUE,
             label = "var", col.var = "black", repel = TRUE,
             legend.title = "Diagnosis") +
  ggtitle("2D PCA-plot from 30 feature dataset") +
  theme(plot.title = element_text(hjust = 0.5))
```

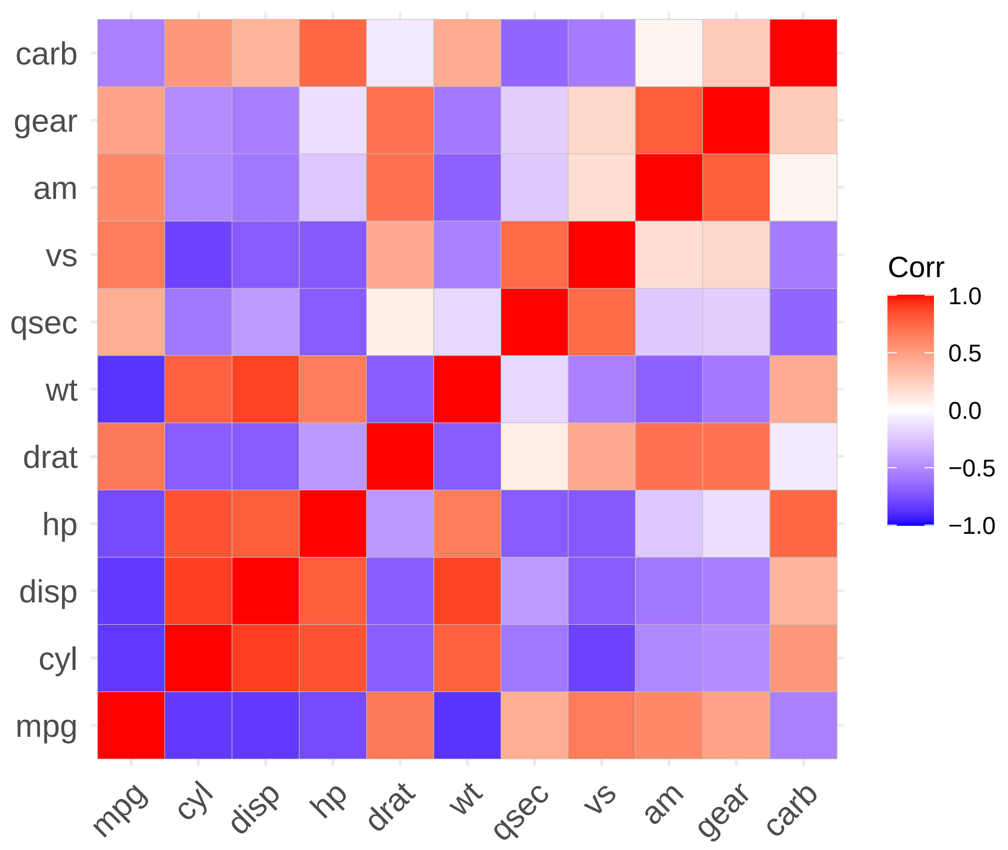


Dimensionality-Reduction-in-R by DataCamp

```
# install.packages("ggcorrplot")
library(ggcorrplot)
data(mtcars)

mtcars$cyl <- as.numeric(as.character(mtcars$cyl))

cor_mtcars <- cor(mtcars, use = "complete.obs")
head(round(cor_mtcars, 2))
```

```
##         mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
## mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
## cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
## disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
## hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
## drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
## wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
```

```
ggcorrplot(cor_mtcars)
```



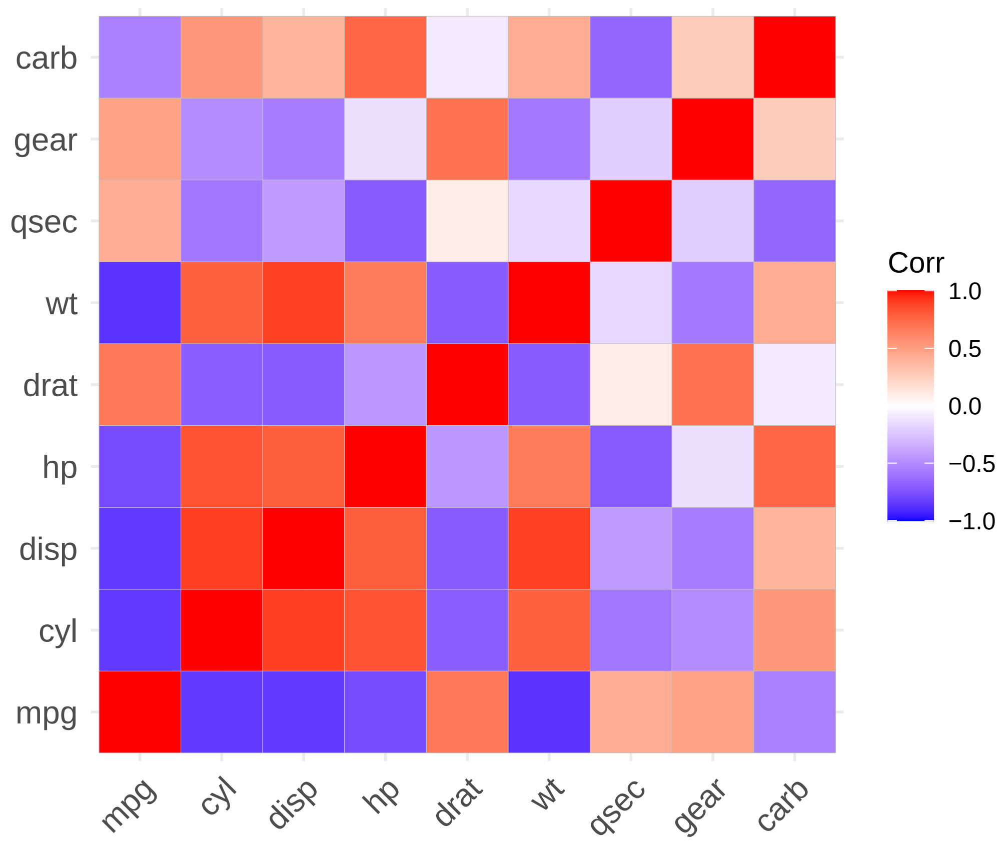
Exploring multivariate data We've loaded a data frame called cars into your workspace. Go ahead and explore it! It includes features of a big range of brands of cars from 2004. In this exercise, you will explore the dataset and attempt to draw useful conclusions from the correlation matrix. Recall that correlation reveals feature resemblance and it will help us infer how cars are related to each other based on their features' values. To this end, you will discover how difficult it is to trace patterns based solely on the correlation structure.

```
# Explore cars with summary()
str(mtcars)
```

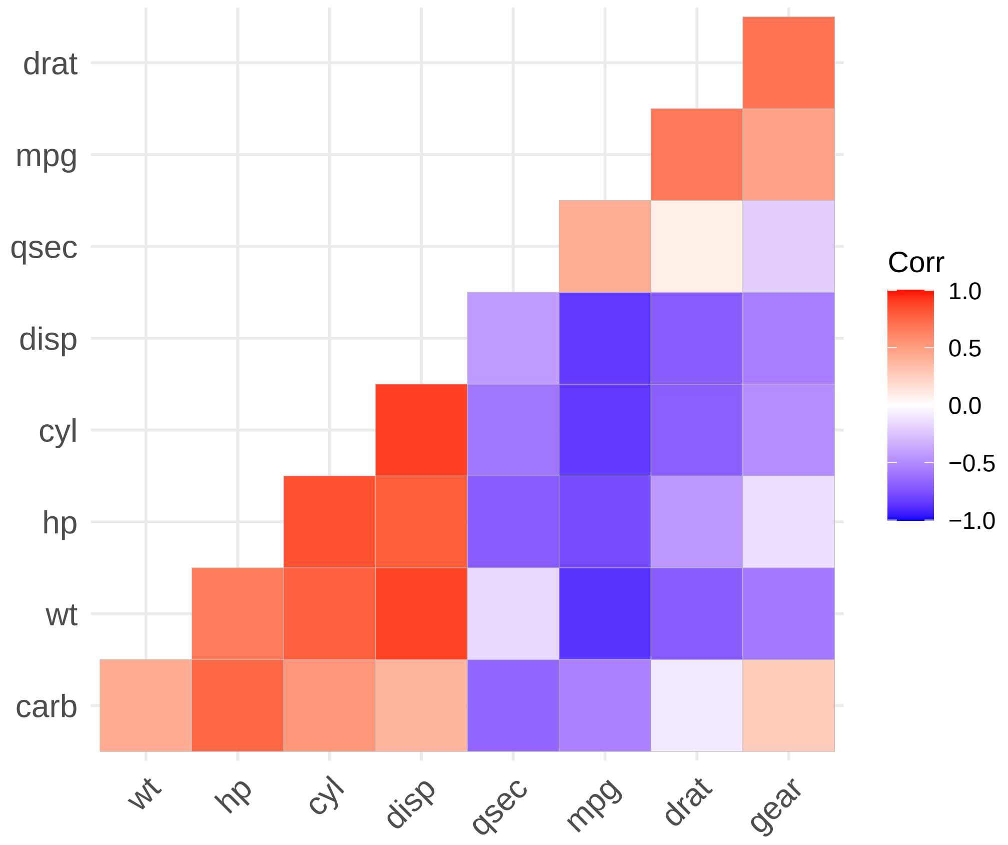
```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
# Get the correlation matrix with cor()
correl <- cor(mtcars[,c(1:7,10,11)], use = "complete.obs")
```

```r
# Use ggcorrplot() to explore the correlation matrix
ggcorrplot(correl)
```



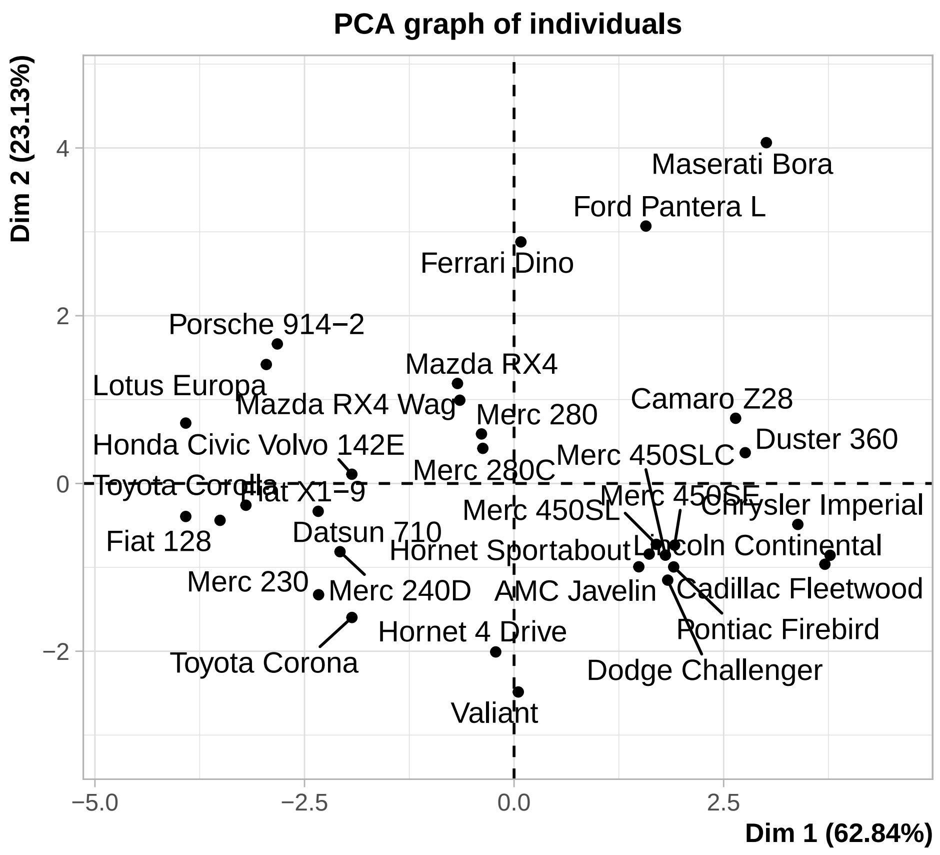
```r
# Conduct hierarchical clustering on the correlation matrix
ggcorrplot_clustered <- ggcorrplot(correl, hc.order = TRUE, type = "lower")
ggcorrplot_clustered
```

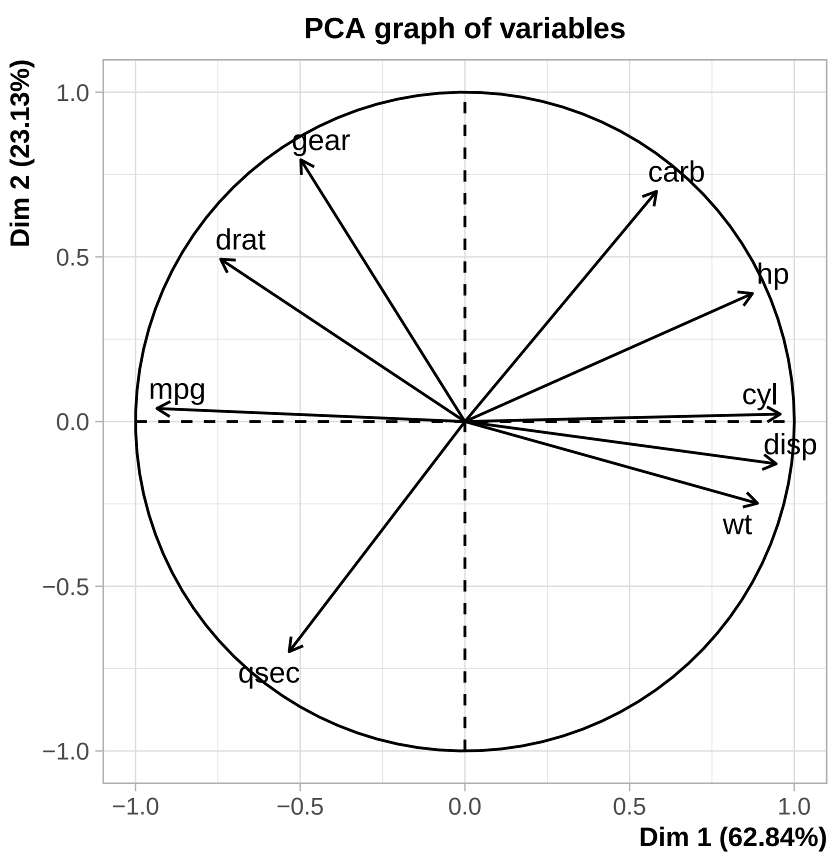


```r
# install.packages("FactoMineR")
library(FactoMineR)

str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```r
mtcars_pca <- PCA(mtcars[,c(1:7, 10,11)], ncp = 6)
```


PCA graph of individuals
Dim 2 (23.13%)
Dim 1 (62.84%)
4
2
0
−2
−5.0
−2.5
0.0
2.5
Maserati Bora
Ford Pantera L
Ferrari Dino
Porsche 914−2
Lotus Europa
Mazda RX4
Mazda RX4 Wag
Merc 280
Camaro Z28
Duster 360
Honda Civic
Volvo 142E
Merc 280C
Merc 450SLC
Toyota Corolla
Fiat X1−9
Merc 450SE
Merc 450SL
Chrysler Imperial
Fiat 128
Datsun 710
Hornet Sportabout
Lincoln Continental
Merc 230
Merc 240D
AMC Javelin
Cadillac Fleetwood
Pontiac Firebird
Hornet 4 Drive
Toyota Corona
Dodge Challenger
Valiant



```
# Names of variables within mtcars_pca
names(mtcars_pca)
```

```
## [1] "eig"  "var"  "ind"  "svd"  "call"
```

```
mtcars_pca$eig
```

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1 5.65593947             62.8437719                          62.84377
## comp 2 2.08210029             23.1344477                          85.97822
## comp 3 0.50421482              5.6023869                          91.58061
## comp 4 0.26502753              2.9447503                          94.52536
## comp 5 0.18315864              2.0350960                          96.56045
## comp 6 0.12379319              1.3754799                          97.93593
## comp 7 0.10506192              1.1673547                          99.10329
## comp 8 0.05851375              0.6501528                          99.75344
## comp 9 0.02219038              0.2465598                         100.00000
```

```
# PC by quality of representation
mtcars_pca$var$cos2
```

```
##          Dim.1        Dim.2        Dim.3        Dim.4       Dim.5        Dim.6
## mpg  0.8742108 0.0015790128 0.0246694081 9.947149e-06 0.018844874 6.420218e-02
## cyl  0.9165420 0.0005138548 0.0321000506 4.390209e-04 0.002513249 6.229512e-03
## disp 0.8930121 0.0164496080 0.0030874485 3.054601e-02 0.043401184 4.792182e-05
## hp   0.7621310 0.1511266410 0.0001493681 1.236360e-03 0.015910403 1.550821e-02
## drat 0.5499243 0.2430363868 0.0113380378 1.895313e-01 0.004802421 2.923669e-05
## wt   0.7889195 0.0615560802 0.1038050787 9.694813e-03 0.006437950 8.687572e-04
## qsec 0.2846821 0.4878338673 0.1989352172 2.437871e-04 0.004025445 8.210137e-03
## gear 0.2481810 0.6316291032 0.0215183006 2.113316e-02 0.057949661 1.291381e-02
## carb 0.3383367 0.4883757368 0.1086119136 1.219310e-02 0.029273449 1.578343e-02
```

```
#PC by contribution
mtcars_pca$var$contrib
```

```
##          Dim.1       Dim.2      Dim.3        Dim.4     Dim.5       Dim.6
## mpg  15.456509  0.07583750  4.8926384  0.003753251 10.288826 51.86244653
## cyl  16.204947  0.02467964  6.3663441  0.165651047  1.372170  5.03219293
## disp 15.788925  0.79004878  0.6123280 11.525599379 23.695953  0.03871119
## hp   13.474879  7.25837471  0.0296239  0.466502562  8.686679 12.52751190
## drat  9.722953 11.67265515  2.2486522 71.513827382  2.622001  0.02361737
## wt   13.948515  2.95644165 20.5874707  3.658039870  3.514959  0.70178106
## qsec  5.033330 23.42989286 39.4544563  0.091985594  2.197792  6.63213935
## gear  4.387972 30.33615172  4.2676850  7.973949823 31.639055 10.43176266
## carb  5.981971 23.45591799 21.5408014  4.600691092 15.982566 12.74983703
```

```
# Most correlated values to PC1
dimdesc(mtcars_pca)
```

```
## $Dim.1
## $quanti
##      correlation      p.value
## cyl    0.9573620 9.987998e-18
## disp   0.9449932 4.195104e-16
## wt     0.8882114 1.186867e-11
## hp     0.8730011 7.238862e-11
## carb   0.5816671 4.798744e-04
## gear  -0.4981777 3.711660e-03
## qsec  -0.5335561 1.662320e-03
## drat  -0.7415688 1.197792e-06
## mpg   -0.9349924 4.804756e-15
##
## attr(,"class")
## [1] "condes" "list "
##
## $Dim.2
## $quanti
##      correlation      p.value
## gear   0.7947510 5.574498e-08
## carb   0.6988388 8.637452e-06
## drat   0.4929872 4.146805e-03
## hp     0.3887501 2.788365e-02
## qsec  -0.6984510 8.780023e-06
##
## attr(,"class")
## [1] "condes" "list "
##
```

```
## $Dim.3
## $quanti
##      correlation    p.value
## qsec   0.4460215 0.01050917
##
## attr(,"class")
## [1] "condes" "list "
##
## $call
## $call$num.var
## [1] 1
##
## $call$proba
## [1] 0.05
##
## $call$weights
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $call$X
##                           Dim.1  mpg cyl  disp  hp drat    wt  qsec gear carb
## Mazda RX4           -0.67485176 21.0   6 160.0 110 3.90 2.620 16.46    4    4
## Mazda RX4 Wag       -0.64739389 21.0   6 160.0 110 3.90 2.875 17.02    4    4
## Datsun 710          -2.33653412 22.8   4 108.0  93 3.85 2.320 18.61    4    1
## Hornet 4 Drive      -0.21874167 21.4   6 258.0 110 3.08 3.215 19.44    3    1
## Hornet Sportabout    1.61236724 18.7   8 360.0 175 3.15 3.440 17.02    3    2
## Valiant              0.05039885 18.1   6 225.0 105 2.76 3.460 20.22    3    1
## Duster 360           2.75782987 14.3   8 360.0 245 3.21 3.570 15.84    3    4
## Merc 240D           -2.07640796 24.4   4 146.7  62 3.69 3.190 20.00    4    2
## Merc 230            -2.33179070 22.8   4 140.8  95 3.92 3.150 22.90    4    2
## Merc 280            -0.38864206 19.2   6 167.6 123 3.92 3.440 18.30    4    4
## Merc 280C           -0.37239188 17.8   6 167.6 123 3.92 3.440 18.90    4    4
## Merc 450SE           1.91482537 16.4   8 275.8 180 3.07 4.070 17.40    3    3
## Merc 450SL           1.69781116 17.3   8 275.8 180 3.07 3.730 17.60    3    3
## Merc 450SLC          1.80535629 15.2   8 275.8 180 3.07 3.780 18.00    3    3
## Cadillac Fleetwood   3.70798697 10.4   8 472.0 205 2.93 5.250 17.98    3    4
## Lincoln Continental  3.76970673 10.4   8 460.0 215 3.00 5.424 17.82    3    4
## Chrysler Imperial    3.38527779 14.7   8 440.0 230 3.23 5.345 17.42    3    4
## Fiat 128            -3.50760397 32.4   4  78.7  66 4.08 2.200 19.47    4    1
## Honda Civic         -3.91645757 30.4   4  75.7  52 4.93 1.615 18.52    4    2
## Toyota Corolla      -3.91656488 33.9   4  71.1  65 4.22 1.835 19.90    4    1
## Toyota Corona       -1.93421725 21.5   4 120.1  97 3.70 2.465 20.01    3    1
## Dodge Challenger     1.83288975 15.5   8 318.0 150 2.76 3.520 16.87    3    2
## AMC Javelin          1.48827419 15.2   8 304.0 150 3.15 3.435 17.30    3    2
## Camaro Z28           2.64298174 13.3   8 350.0 245 3.73 3.840 15.41    3    4
## Pontiac Firebird     1.90423467 19.2   8 400.0 175 3.08 3.845 17.05    3    2
## Fiat X1-9           -3.19868255 27.3   4  79.0  66 4.08 1.935 18.90    4    1
## Porsche 914-2       -2.82386873 26.0   4 120.3  91 4.43 2.140 16.70    5    2
## Lotus Europa        -2.95550050 30.4   4  95.1 113 3.77 1.513 16.90    5    2
## Ford Pantera L       1.57289857 15.8   8 351.0 264 4.22 3.170 14.50    5    4
## Ferrari Dino         0.08178803 19.7   6 145.0 175 3.62 2.770 15.50    5    6
## Maserati Bora        3.00993147 15.0   8 301.0 335 3.54 3.570 14.60    5    8
## Volvo 142E          -1.93490923 21.4   4 121.0 109 4.11 2.780 18.60    4    2
```

####PCA with FactoMineR As you saw in the video, FactoMineR is a very useful package, rich in functionality, that implements a number of dimensionality reduction methods. Its function for doing PCA is PCA() - easy to remember! Recall that PCA(), by default, generates 2 graphs and extracts the first 5 PCs. **You can use the ncp argument to manually set the number of dimensions to keep.**

You can also use the summary() function to get a quick overview of the indices of the first three principal components. Moreover, for extracting summaries of some of the rows in a dataset, you can specify the nbelements argument. You'll have a chance to practice all of these and more in this exercise!

As in the previous lesson, the cars dataset is available in your workspace.

```
# Run a PCA for the 10 non-binary numeric variables of cars
# The graph argument is set to F to suppress the default graphical output PCA().
pca_output_ten_v <- PCA(mtcars[,c(1:7, 10,11)], ncp = 6, graph = FALSE)
pca_output_ten_v
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 32 individuals, described by 9 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$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

```
# Get the summary of the first 100 cars.
summary(pca_output_ten_v, nbelements = 20)
```

```
##
## Call:
## PCA(X = mtcars[, c(1:7, 10, 11)], ncp = 6, graph = FALSE)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               5.656   2.082   0.504   0.265   0.183   0.124   0.105
## % of var.             62.844  23.134   5.602   2.945   2.035   1.375   1.167
## Cumulative % of var.  62.844  85.978  91.581  94.525  96.560  97.936  99.103
##                        Dim.8   Dim.9
## Variance               0.059   0.022
## % of var.              0.650   0.247
## Cumulative % of var.  99.753 100.000
##
```

```
## Individuals (the 20 first)
##                         Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
## Mazda RX4           |  1.658 | -0.675  0.252  0.166 |  1.192  2.133  0.517 |
## Mazda RX4 Wag       |  1.446 | -0.647  0.232  0.200 |  0.993  1.479  0.471 |
## Datsun 710          |  2.485 | -2.337  3.016  0.884 | -0.332  0.165  0.018 |
## Hornet 4 Drive      |  2.094 | -0.219  0.026  0.011 | -2.008  6.054  0.920 |
## Hornet Sportabout   |  2.138 |  1.612  1.436  0.569 | -0.842  1.064  0.155 |
## Valiant             |  2.667 |  0.050  0.001  0.000 | -2.486  9.275  0.869 |
## Duster 360          |  2.949 |  2.758  4.202  0.875 |  0.367  0.202  0.015 |
## Merc 240D           |  2.470 | -2.076  2.382  0.707 | -0.813  0.993  0.108 |
## Merc 230            |  3.459 | -2.332  3.004  0.454 | -1.326  2.641  0.147 |
## Merc 280            |  1.290 | -0.389  0.083  0.091 |  0.590  0.523  0.209 |
## Merc 280C           |  1.443 | -0.372  0.077  0.067 |  0.419  0.263  0.084 |
## Merc 450SE          |  2.143 |  1.915  2.026  0.799 | -0.736  0.812  0.118 |
## Merc 450SL          |  1.970 |  1.698  1.593  0.743 | -0.726  0.791  0.136 |
## Merc 450SLC         |  2.094 |  1.805  1.801  0.744 | -0.855  1.096  0.167 |
## Cadillac Fleetwood  |  3.989 |  3.708  7.597  0.864 | -0.963  1.393  0.058 |
## Lincoln Continental |  4.035 |  3.770  7.852  0.873 | -0.856  1.100  0.045 |
## Chrysler Imperial   |  3.638 |  3.385  6.332  0.866 | -0.488  0.358  0.018 |
## Fiat 128            |  3.624 | -3.508  6.798  0.937 | -0.440  0.290  0.015 |
## Honda Civic         |  4.235 | -3.916  8.475  0.855 |  0.720  0.778  0.029 |
## Toyota Corolla      |  4.056 | -3.917  8.475  0.932 | -0.393  0.232  0.009 |
##                       Dim.3    ctr   cos2
## Mazda RX4           -0.208  0.267  0.016 |
## Mazda RX4 Wag        0.113  0.079  0.006 |
## Datsun 710          -0.214  0.283  0.007 |
## Hornet 4 Drive      -0.335  0.694  0.026 |
## Hornet Sportabout   -1.050  6.827  0.241 |
## Valiant              0.114  0.080  0.002 |
## Duster 360          -0.662  2.720  0.050 |
## Merc 240D            0.863  4.611  0.122 |
## Merc 230             2.000 24.791  0.334 |
## Merc 280             0.901  5.026  0.487 |
## Merc 280C            1.167  8.440  0.654 |
## Merc 450SE          -0.209  0.272  0.010 |
## Merc 450SL          -0.332  0.682  0.028 |
## Merc 450SLC         -0.087  0.047  0.002 |
## Cadillac Fleetwood   0.897  4.990  0.051 |
## Lincoln Continental  0.947  5.561  0.055 |
## Chrysler Imperial    0.681  2.877  0.035 |
## Fiat 128            -0.230  0.327  0.004 |
## Honda Civic         -0.230  0.329  0.003 |
## Toyota Corolla      -0.259  0.416  0.004 |
##
## Variables
##                        Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## mpg                 | -0.935 15.457  0.874 |  0.040  0.076  0.002 | -0.157
## cyl                 |  0.957 16.205  0.917 |  0.023  0.025  0.001 | -0.179
## disp                |  0.945 15.789  0.893 | -0.128  0.790  0.016 | -0.056
## hp                  |  0.873 13.475  0.762 |  0.389  7.258  0.151 | -0.012
## drat                | -0.742  9.723  0.550 |  0.493 11.673  0.243 |  0.106
## wt                  |  0.888 13.949  0.789 | -0.248  2.956  0.062 |  0.322
## qsec                | -0.534  5.033  0.285 | -0.698 23.430  0.488 |  0.446
## gear                | -0.498  4.388  0.248 |  0.795 30.336  0.632 |  0.147
## carb                | 0.582  5.982  0.338 |  0.699 23.456  0.488 |  0.330
##                       ctr   cos2
## mpg                  4.893  0.025 |
## cyl                  6.366  0.032 |
## disp                 0.612  0.003 |
## hp                   0.030  0.000 |
## drat                 2.249  0.011 |
## wt                  20.587  0.104 |
## qsec                39.454  0.199 |
## gear                 4.268  0.022 |
## carb                21.541  0.109 |
```

```
# Get the variance of the first 3 new dimensions.
pca_output_ten_v$eig[,2][1:3]
```

```
##    comp 1    comp 2    comp 3
## 62.843772 23.134448  5.602387
```

```
# Get the cumulative variance.
pca_output_ten_v$eig[,3][1:3]
```

```
##   comp 1   comp 2   comp 3
## 62.84377 85.97822 91.58061
```

Exploring PCA() PCA() provides great flexibility in its usage. You can choose 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.

Your learning objectives in this exercise are:
- To conduct PCA considering parts of a dataset - To inspect the most correlated variables with a specified principal component - To find the contribution of variables in the designation of the first 5 principal components

Go for it! The cars dataset is available in your workspace.

```
# Run a PCA on cars by setting the first 8 variables as
# quantitative supplementary variables and the last 2
# variables as qualitative supplementary ones. The graph
# argument is set to F for not displaying the default graphical output.

# Run a PCA with active and supplementary variables

# pca_output_all <- PCA(cars, quanti.sup = 1:8, quali.sup = 20:21, graph = FALSE)

# Get the most correlated variables
# inspect the most correlated variables with
# the first two principal components by specifying the axes argument.

# dimdesc(pca_output_all, axes = 1:2)

# Run a PCA on the first 100 car categories
# pca_output_hundred <- PCA(cars, quanti.sup = 1:8, quali.sup = 20:21, ind.sup = 101:nrow(cars), graph

# Trace variable contributions in pca_output_hundred
# pca_output_hundred$var$contrib
```

PCA with ade4 Alright! Now that you've got some real hands-on experience with FactoMineR, let's have a look at ade4, a well-known and well-maintained R package with a large number of numerical methods for building and handling PCA models.

dudi.pca() is the main function that implements PCA for ade4 and by default, it is interactive: It lets the user insert the number of retained dimensions. For suppressing the interactive mode and inserting the number of axes within the dudi.pca() function, you need to set the scannf argument to FALSE and then use the nf argument for setting the number of axes to retain. So, let's put ade4 into practice and compare it with FactoMineR.

```
# install.packages("ade4")
library(ade4)
```

```
##
## Attaching package: 'ade4'

## The following object is masked from 'package:FactoMineR':
##
##     reconst
```

```
# Run a PCA using the 10 non-binary numeric variables.
cars_pca <- dudi.pca(mtcars[,c(1:7, 10,11)], scannf = F, nf = 4)

# Explore the summary of cars_pca.
summary(cars_pca)
```

```
## Class: pca dudi
## Call: dudi.pca(df = mtcars[, c(1:7, 10, 11)], scannf = F, nf = 4)
##
## Total inertia: 9
##
## Eigenvalues:
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  5.6559  2.0821  0.5042  0.2650  0.1832
##
## Projected inertia (%):
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  62.844  23.134   5.602   2.945   2.035
##
## Cumulative projected inertia (%):
##     Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
##   62.84   85.98   91.58   94.53   96.56
##
## (Only 5 dimensions (out of 9) are shown)
```

```r
# Explore the summary of pca_output_ten_v.
pca_output_ten_v
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 32 individuals, described by 9 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$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Plotting cos2 You're getting the hang of PCA now! As Alex demonstrated in the video, an important index included in your PCA models is the squared cosine, abbreviated in FactoMineR and factoextra as cos2. This shows how accurate the representation of your variables or individuals on the PC plane is.

The factoextra package is excellent at handling PCA models built using FactoMineR. Here, you're going to explore the functionality of factoextra. You'll be using the pca_output_all object that you computed earlier and create plots based on its cos2. Visual aids are key to understanding cos2.

```r
# Create a factor map for the variables.
# fviz_pca_var(pca_output_all, select.var = list(cos2 = 0.7), repel = TRUE)

# Create a barplot for the variables with the highest cos2 in the 2nd PC.
# fviz_cos2(pca_output_all, choice = "var", axes = 2, top = 10)
```

Plotting contributions In this exercise, you will be asked to prepare a number of plots to help you get a better feeling of the variables' contributions on the extracted principal components. It is important to keep in mind that the contributions of the variables essentially signify their importance for the construction of a given principal component.

```r
# Create a factor map for the top 5 variables with the highest contributions.
# fviz_pca_var(pca_output_all, select.var = list(contrib = 5), repel = TRUE)

# Create a factor map for the top 5 individuals with the highest contributions.
# fviz_pca_ind(pca_output_all, select.ind = list(contrib = 5), repel = TRUE)

# Create a barplot for the variables with the highest contributions to the 1st PC.
# fviz_contrib(pca_output_all, choice = "var", axes = 1, top = 5)
# Create a barplot for the variables with the highest contributions to the 2nd PC.
# fviz_contrib(pca_output_all, choice = "var", axes = 2, top = 5)
```

Biplots and their ellipsoids As mentioned in the video, biplots are graphs that provide a compact way of summarizing the relationships between individuals, variables, and also between variables and individuals within the same plot. Moreover, ellipsoids can be added on top of a biplot and offer a much better overview of the biplot based on the groupings of variables and individuals.

In this exercise, your job is to create biplots and ellipsoids using factoextra's graphical utilities.

```
# Create a biplot with no labels for all individuals with the geom argument.
# fviz_pca_biplot(pca_output_all)

# Create ellipsoids for wheeltype columns respectively.
# fviz_pca_ind(pca_output_all, habillage = cars$wheeltype, addEllipses = TRUE)

# Create the biplot with ellipsoids
# fviz_pca_biplot(pca_output_all, habillage = cars$wheeltype, addEllipses = TRUE, alpha.var = "cos2")
```

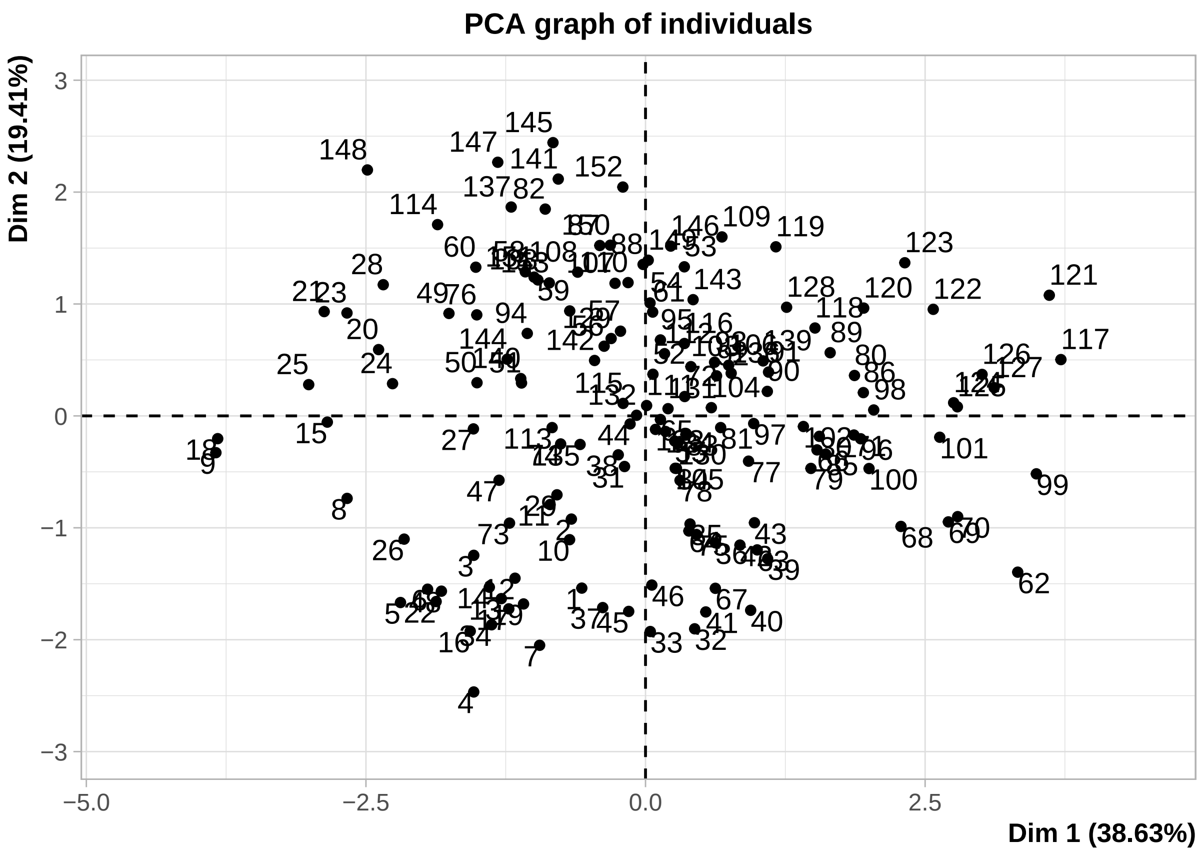
The Kaiser-Guttman rule and the Scree test In the video, you saw the three most common methods that people utilize to decide the number of principal components to retain:

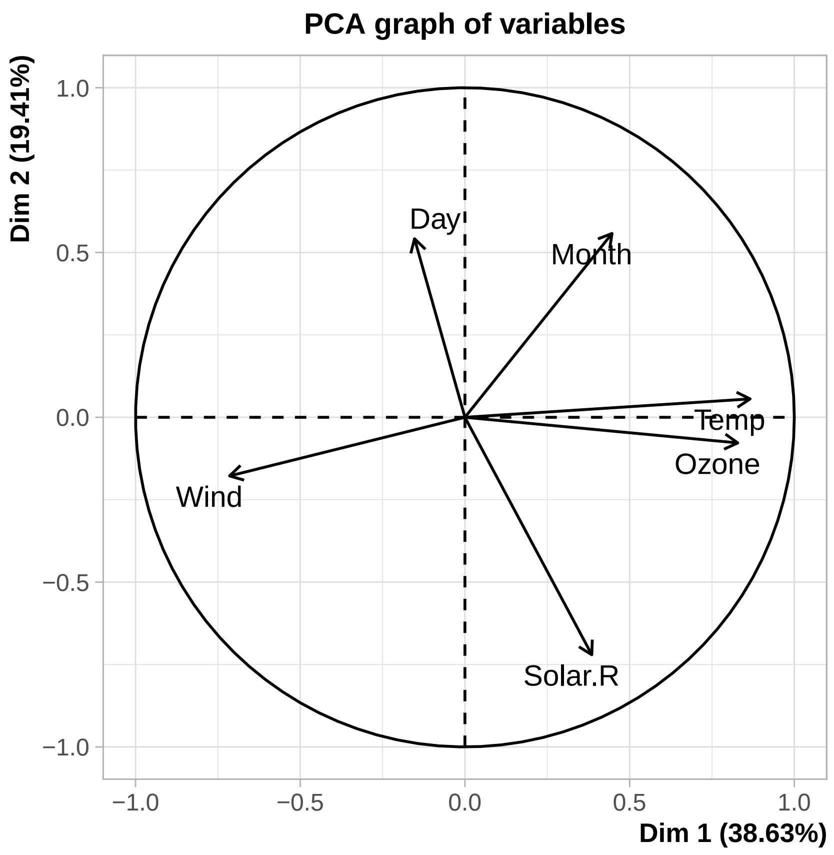
- Scree test (constructing the screeplot)
- The Kaiser-Guttman rule
- Parallel Analysis

Your task now is to apply all of them on the R's built-in airquality dataset!

```
# Conduct a PCA on the airquality dataset
pca_air <- PCA(airquality)
```

```
## Warning in PCA(airquality): Missing values are imputed by the mean of the
## variable: you should use the imputePCA function of the missMDA package
```

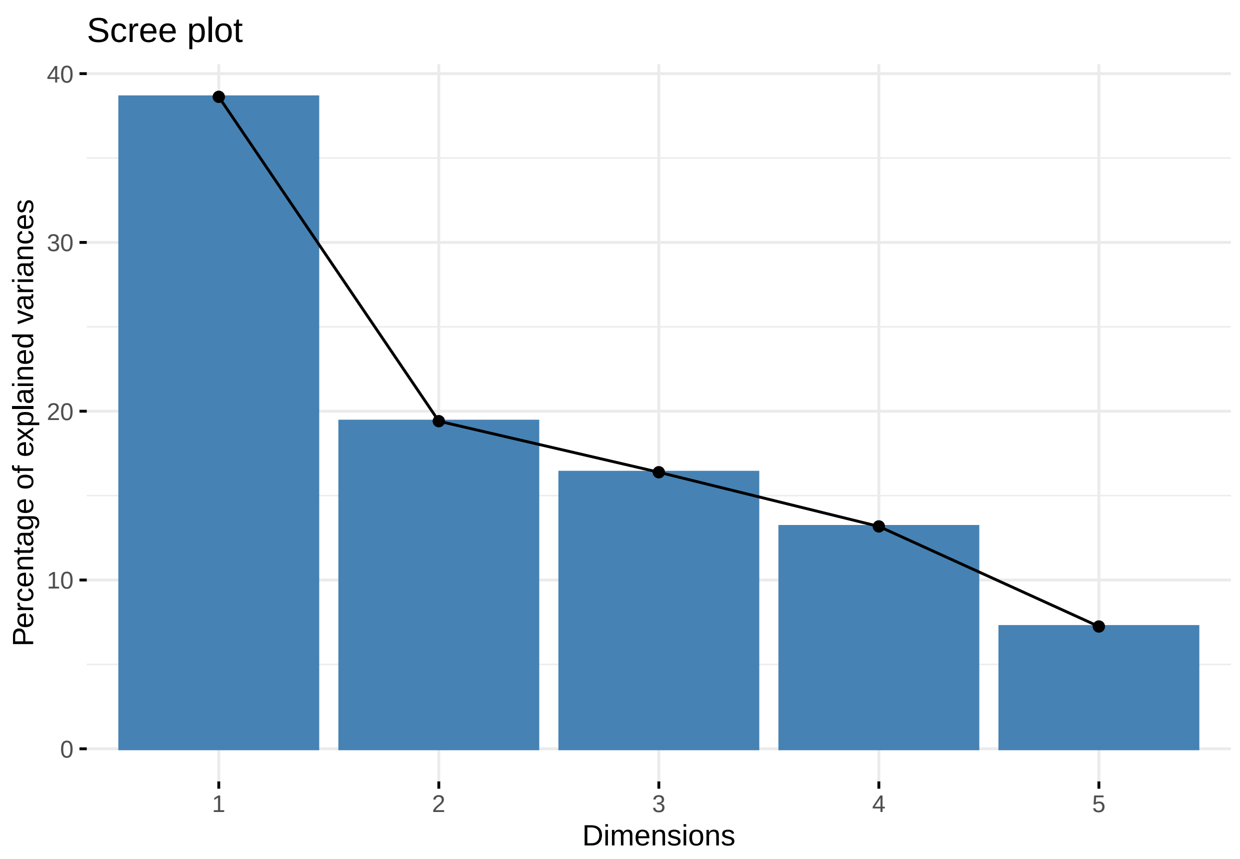

PCA graph of individuals
Dim 1 (38.63%)
Dim 2 (19.41%)



```r
# Apply the Kaiser-Guttman rule
summary(pca_air, ncp = 4)
```

```
##
## Call:
## PCA(X = airquality)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6
## Variance               2.318   1.165   0.983   0.790   0.435   0.310
## % of var.             38.625  19.411  16.385  13.175   7.246   5.158
## Cumulative % of var.  38.625  58.036  74.421  87.596  94.842 100.000
##
## Individuals (the 10 first)
##               Dist     Dim.1    ctr   cos2     Dim.2    ctr   cos2     Dim.3    ctr
## 1         |  2.582 | -0.570  0.092  0.049 | -1.539  1.329  0.355 | -0.229  0.035
## 2         |  2.404 | -0.663  0.124  0.076 | -0.922  0.477  0.147 | -0.437  0.127
## 3         |  2.473 | -1.536  0.665  0.386 | -1.246  0.871  0.254 | -0.834  0.463
## 4         |  3.101 | -1.536  0.665  0.245 | -2.467  3.416  0.633 | -0.148  0.015
## 5         |  3.225 | -2.191  1.354  0.462 | -1.668  1.561  0.267 | -0.136  0.012
## 6         |  2.653 | -1.948  1.071  0.540 | -1.549  1.346  0.341 | -0.368  0.090
## 7         |  2.667 | -0.947  0.253  0.126 | -2.050  2.358  0.591 |  0.257  0.044
## 8         |  3.101 | -2.668  2.008  0.741 | -0.737  0.305  0.057 | -0.302  0.060
## 9         |  4.380 | -3.841  4.161  0.769 | -0.329  0.061  0.006 | -0.874  0.508
## 10        |  1.863 | -0.679  0.130  0.133 | -1.106  0.687  0.353 |  0.455  0.137
##            cos2    Dim.4    ctr   cos2
## 1         0.008 | -1.861  2.863  0.519 |
## 2         0.033 | -2.072  3.551  0.743 |
## 3         0.114 | -1.001  0.828  0.164 |
## 4         0.002 | -0.318  0.084  0.011 |
## 5         0.002 | -0.782  0.506  0.059 |
## 6         0.019 | -0.445  0.163  0.028 |
## 7         0.009 | -0.696  0.400  0.068 |
## 8         0.009 | -1.140  1.074  0.135 |
## 9         0.040 | -0.526  0.229  0.014 |
## 10        0.060 | -1.207  1.205  0.420 |
##
## Variables
##             Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
## Ozone   |  0.828 29.610  0.686 | -0.078  0.517  0.006 |  0.295  8.877  0.087 |
## Solar.R |  0.385  6.402  0.148 | -0.720 44.559  0.519 |  0.167  2.821  0.028 |
## Wind    | -0.715 22.029  0.511 | -0.178  2.719  0.032 | -0.200  4.072  0.040 |
## Temp    |  0.866 32.341  0.750 |  0.056  0.267  0.003 | -0.126  1.612  0.016 |
## Month   |  0.447  8.608  0.199 |  0.558 26.725  0.311 | -0.514 26.881  0.264 |
## Day     | -0.153  1.010  0.023 |  0.542 25.212  0.294 |  0.740 55.737  0.548 |
##           Dim.4    ctr   cos2
## Ozone    -0.082  0.854  0.007 |
## Solar.R   0.481 29.245  0.231 |
## Wind      0.493 30.782  0.243 |
## Temp      0.127  2.039  0.016 |
## Month     0.404 20.665  0.163 |
## Day       0.360 16.416  0.130 |
```

```
# Perform the screeplot test
fviz_screeplot(pca_air, ncp = 5)
```



Parallel Analysis with paran() In this exercise, you will use two R functions for conducting parallel analysis for PCA:

- paran() of the paran package and
- fa.parallel() of the psych package.

fa.parallel() has one advantage over the paran() function; it allows you to use more of your data while building the correlation matrix. On the other hand, paran() does not handle missing data and you should first exclude missing values before passing the data to the function. For checking out the suggested number of PCs to retain, fa.parallel()'s output object includes the attribute ncomp.

The built-in R dataset airquality, on which you will be doing your parallel analyses, describes daily air quality measurements in New York from May to September 1973 and includes missing values.

```
# install.packages("paran")
# install.packages("psych")

library(paran)
```

```
## Loading required package: MASS
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
## The following objects are masked from 'package:scales':
##
##     alpha, rescale

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

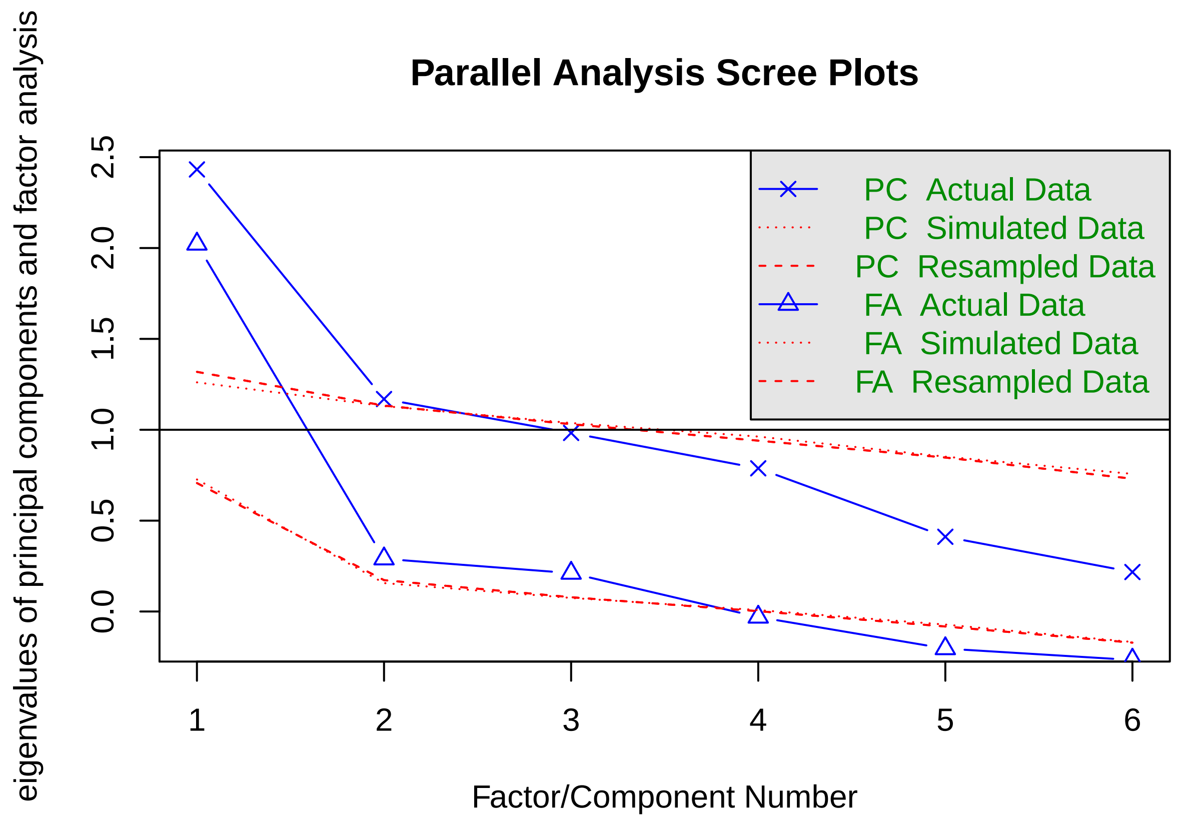
```
# Subset the complete rows of airquality.
airquality_complete <- airquality[complete.cases(airquality), ]

# Conduct a parallel analysis with paran().
air_paran <- paran(airquality_complete, seed = 1)
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 180 iterations, using the mean estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           2.137510    2.468840      0.331329
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (1 components retained)
```

```
# Check out air_paran's suggested number of PCs to retain.
air_paran$Retained
```

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

```
# Conduct a parallel analysis with fa.parallel().
air_fa_parallel <- fa.parallel(airquality)
```



```
## Parallel analysis suggests that the number of factors =  3  and the number of components =  1
```

```
# Check out air_fa_parallel's suggested number of PCs to retain.
air_fa_parallel$ncomp
```

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