Lab 4 - PCA\_1

Table of contents

## Lab 4

Today we are going to explore “Principle Component Analysis”.

## Principal Component Analysis

### Data set

We’ll use the built in dataset mtcars, which contains information on a number of cars from the magazine Motor Trend. For each car, the dataset gives the number of miles per gallon (mpg), number of cylinders (cyl), displacement in cubic centimetres (disp), and many other variables.

A complete list of the variables is given in the R documentation. Use head() to view the columns:

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

Two columns represent categorical data (engine vs and transmission am). These columns have entries 0 or 1, representing V-shaped or straight engine, and automatic or manual transmission, respectively.

A general warning: some data sets might codify categorical variables making them look like numeric. Pay attention to your data!

Remove these columns so that we only use continuous data. Usetwo ways to do this, the dplyr way:

mtcars <- dplyr::select(mtcars, !vs & !am)

#### Exercise (Select using Base R)

How would you exclude columns vs and am without using dplyr? You can simply use dplyr, but attempting this in base R gives you a glimpse into the inner working of dplyr. Hint: see ?which.

#mtcars <- mtcars[, c(1:7, 10, 11)]  
# If you want to use negative subsetting:  
#mtcars <- mtcars[, -c(8, 9)]

Warning: doing this will overwrite the mtcars dataset (at least until the next session). If you want to retain the original data set, name the trimmed data frame differently.

Note: R begins indexing at 1, not 0.

### Princomp

R has two built in functions to perform PCA. For this class we will use princomp. To centre and scale the data, the optional argument cor is set to TRUE.

mtcars.pca <- princomp(mtcars, cor = TRUE)

The function summary() displays the principal components ranked by proportion of variance captured:

summary(mtcars.pca)

Importance of components:  
 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5  
Standard deviation 2.3782219 1.4429485 0.71008086 0.5148082 0.42797037  
Proportion of Variance 0.6284377 0.2313445 0.05602387 0.0294475 0.02035096  
Cumulative Proportion 0.6284377 0.8597822 0.91580607 0.9452536 0.96560453  
 Comp.6 Comp.7 Comp.8 Comp.9  
Standard deviation 0.3518426 0.32413257 0.241896155 0.148964367  
Proportion of Variance 0.0137548 0.01167355 0.006501528 0.002465598  
Cumulative Proportion 0.9793593 0.99103287 0.997534402 1.000000000

**Q: Explain to a lay person (e.g. Boris Johnson) what the summary table above tells us. That is, what does Comp.1-9 mean? What does Standard deviation, Proportion of Variance and Cumulative Proportion mean?**

*A: The table derived from* ***mtcars.pca*** *depicts data from the* ***mtcars*** *dataset simplified into nine principal components (Comp.1 to Comp.9). Each component represents a bundle of car features, with Comp.1 capturing the most significant traits that differentiate the cars, and Comp.9 the least.*

* ***Standard Deviation****: It shows how much the car models differ in each component. Higher values mean greater variation in those features among the models.*
* ***Proportion of Variance****: This indicates the percentage of total variation each component represents. For example, if Comp.1 has a high proportion, the features it encapsulates are crucial in distinguishing the car models.*
* ***Cumulative Proportion****: This cumulatively adds up the proportion of variance from Comp.1 to Comp.9, providing insight into how much of the total data’s variation is represented as you move through the components.*

*In simple terms, the table helps to understand and compare car models efficiently, highlighting the most distinguishing features and showing how much each set of features contributes to the overall variation among models.*

### PCA Scores and Loadings

The output of princomp is a **list** containing various relevant outputs of the PCA procedure. To quickly view the list elements use:

str(mtcars.pca)

List of 7  
 $ sdev : Named num [1:9] 2.378 1.443 0.71 0.515 0.428 ...  
 ..- attr(\*, "names")= chr [1:9] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...  
 $ loadings: 'loadings' 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] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...  
 $ 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] 5.932 1.758 121.987 67.483 0.526 ...  
 ..- attr(\*, "names")= chr [1:9] "mpg" "cyl" "disp" "hp" ...  
 $ n.obs : int 32  
 $ scores : num [1:32, 1:9] 0.675 0.647 2.337 0.219 -1.612 ...  
 ..- attr(\*, "dimnames")=List of 2  
 .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...  
 .. ..$ : chr [1:9] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...  
 $ call : language princomp(x = mtcars, cor = TRUE)  
 - attr(\*, "class")= chr "princomp"

This can be important for interpreting results of PCA!

**Q: I have put the following question *“Explain again to our lay person what each of the list of the 7 outputs mean from above.”* into ChatGPT, which gives us the following explanations. I want you to look up the documentation to check if the descriptions below are correct - if not, how would you describe them?**

1. **sdev (Standard Deviations):** These are like the speedometers for each trait of the cars, showing how much the models differ for specific characteristics. A higher ‘speed’ (or deviation) means that a characteristic varies significantly among the car models.
2. **loadings (Loadings):** Loadings are like the contribution of each car part to its overall performance. For instance, how much do the engine, tires, and brakes each contribute to a car’s speed? Loadings help us understand the weight of each feature in the components.
3. **center (Center):** Think of the center as the ‘average car’ in our dataset. It represents the baseline from which we measure and compare the other cars, considering all their features.
4. **scale (Scale):** The scale is like the measuring tape we use. It ensures that we fairly compare all cars regardless of their original units, like comparing the weight and size of different car models.
5. **n.obs (Number of Observations):** This is the total number of car models we examined or observed in our dataset.
6. **scores (Scores):** Scores are like individual performance reports for each car, based on the traits (components) we’ve identified. It tells us how each car model performs concerning each trait.
7. **call (Call):** This is the specific command or ‘instruction manual’ we used to generate this analysis. It helps anyone who reads the report to understand how the analysis was conducted.

### Biplots

We can visualise the results in a biplot (see Week 4 slides). R does not have a built-in biplot function so we install ggbiplot as follows:

#install.packages("devtools", dependencies = TRUE)  
library("devtools")

Warning: package 'devtools' was built under R version 4.2.3

Loading required package: usethis

#install\_github("vqv/ggbiplot")  
library(ggbiplot)

Loading required package: ggplot2

Warning: package 'ggplot2' was built under R version 4.2.3

Loading required package: plyr

Loading required package: scales

Warning: package 'scales' was built under R version 4.2.3

Loading required package: grid

The default package repository is CRAN. Some experimental packages might be available on Github instead of CRAN. The function install\_github from devtools allows direct installation from github.

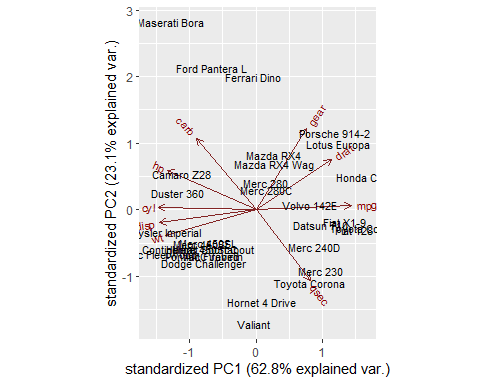
Packages only need to be installed once, and then imported in each session. Optional alternative base R methods are provided at the end of these slides.

**Q: What is github and who owns it? Again, try to explain this to a lay person.**

*A: GitHub is like a huge online library but for computer code instead of books. People around the world who write code, either for work, school, or as a hobby, can use GitHub to store their code safely. They can also share it with others, collaborate with teammates, and even use it to showcase their work to the public or potential employers. It is owned by Microsoft.*

Create a biplot by calling ggbiplot on mtcars.pca, and display the names of each car using labels.

ggbiplot(mtcars.pca, labels = rownames(mtcars))



**Q: How can we make this figure larger? Save this image and specify the dimensions.**

### Interpreting Results

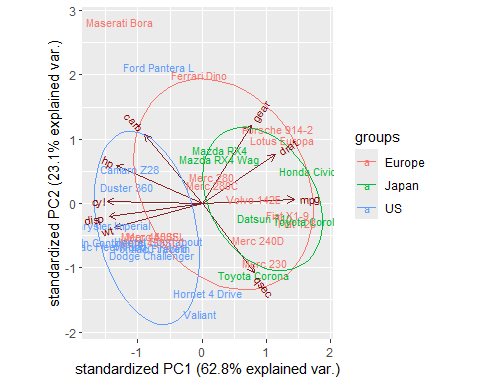
A reasonable assumption about the cars is that they might have similar properties if they are produced in the same region.

We can find evidence for (or against) this hypothesis by finding out where each car was manufactured, and color-coding points in the biplot. Create vector of countries of origin for each car

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))

The groups argument in ggbiplot assigns each group a color, so we can color-code cars based on their region using groups = mtcars.country.Setting ellipse = TRUE draws a shape around points in the same group, which will make clusters easier to see.

ggbiplot(mtcars.pca, ellipse = TRUE, labels = rownames(mtcars), groups = mtcars.country)



## Optional Content: Base R Implementation

library(dplyr)

Warning: package 'dplyr' was built under R version 4.2.3

Attaching package: 'dplyr'

The following objects are masked from 'package:plyr':  
  
 arrange, count, desc, failwith, id, mutate, rename, summarise,  
 summarize

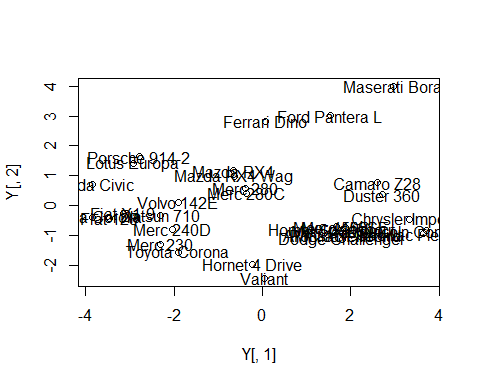
The following objects are masked from 'package:stats':  
  
 filter, lag

The following objects are masked from 'package:base':  
  
 intersect, setdiff, setequal, union

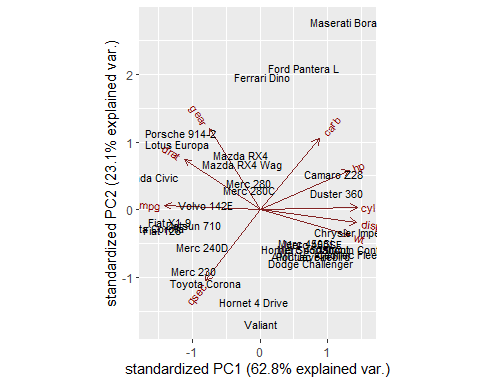
#### Base R implementation ####  
mtcars\_std <- lapply(mtcars, function(x) {  
 (x - mean(x))/sd(x)  
}) %>%  
 list2DF()  
mtcars\_svd <- svd(mtcars\_std)  
mtcars\_V <- mtcars\_svd$v  
Y <- as.matrix(mtcars\_std) %\*% mtcars\_V  
  
# Compare against mtcars  
unclass(mtcars.pca$loadings/mtcars\_V) %>%  
 head()

Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9  
mpg -1 1 -1 -1 -1 1 -1 -1 1  
cyl -1 1 -1 -1 -1 1 -1 -1 1  
disp -1 1 -1 -1 -1 1 -1 -1 1  
hp -1 1 -1 -1 -1 1 -1 -1 1  
drat -1 1 -1 -1 -1 1 -1 -1 1  
wt -1 1 -1 -1 -1 1 -1 -1 1

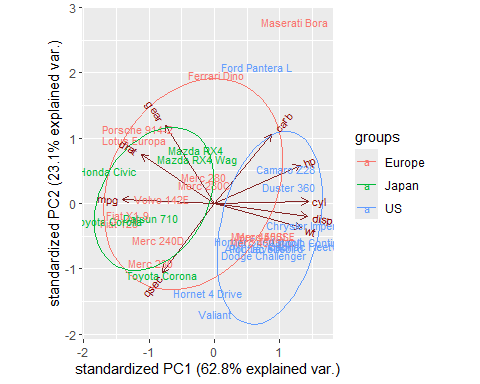
plot(Y[, 1], Y[, 2])  
text(Y[, 1], Y[, 2], labels = rownames(mtcars))



mt\_pr <- prcomp(mtcars, scale. = T)  
ggbiplot(mt\_pr, labels = rownames(mtcars))



ggbiplot(mt\_pr, ellipse = TRUE, labels = rownames(mtcars), groups = mtcars.country)



#ggcorrplot(cor(mtcars)) #fviz\_eig(mtcars.pca, addlabels = TRUE) #fviz\_cos2(mtcars.pca, choice = "var", axes = 1) #fviz\_cos2(mtcars.pca, choice = "var", axes = 2) #fviz\_cos2(mtcars.pca, choice = "var", axes = 3)