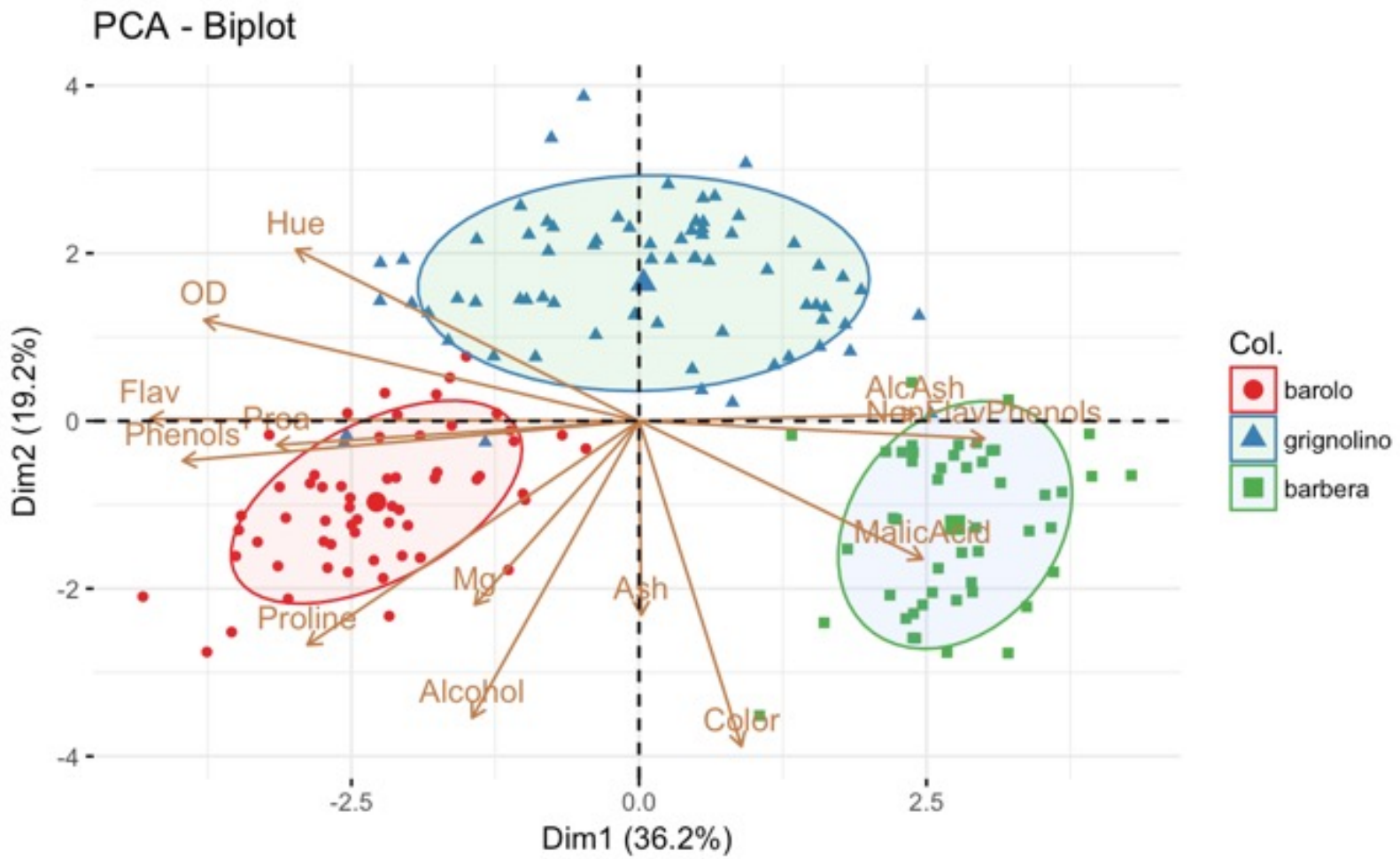
Lecture 8: Multivariate Analysis

Jing Ma, Statistics, TAMU

28 October, 2019

Multivariate Analysis



This lecture

- Data matrices
  - Data summaries and visualizations
  - Preprocessing the data
- Dimension reduction
  - Projecting 2D data on a line
  - Principal component analysis (PCA)

What is multivariate analysis?

- Most of datasets consist of measurements of *multiple variables* repeated for a number of observations.

What is multivariate analysis?

- Most of datasets consist of measurements of *multiple variables* repeated for a number of observations.
- Modern datasets contain thousands to millions of dimensions.

What is multivariate analysis?

- Most of datasets consist of measurements of *multiple variables* repeated for a number of observations.
- Modern datasets contain thousands to millions of dimensions.
- Multivariate analysis (MVA) is a technique to analyze more than one variable at a time.

What is multivariate analysis?

- Most of datasets consist of measurements of *multiple variables* repeated for a number of observations.
- Modern datasets contain thousands to millions of dimensions.
- Multivariate analysis (MVA) is a technique to analyze more than one variable at a time.
- MVA encompasses many statistical methods.
  - PCA
  - Multidimensional scaling
  - Correspondence analysis
  - Factor analysis
  - MANOVA (and many others)

Why do you need multivariate analysis?

- In a lot of cases, you are not interested in how a single variable behaves on its own, but how all variables or a group of variables behave together jointly or how they affect each other.
- The main purpose of MVA is to **investigate connections or associations** between different measured variables.
- **If all variables in a dataset are independent (unrelated), then we should just study each column separately** and use standard univariate statistics on each one by one.

Data in a matrix format

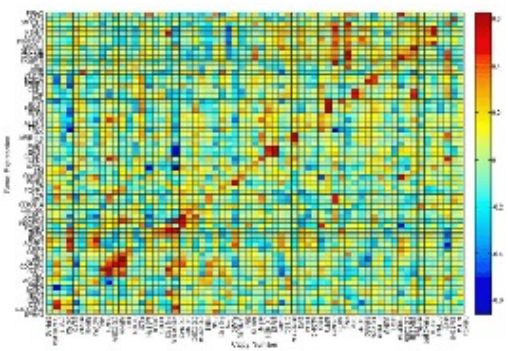
- Datasets can often be organized in a form of a rectangular matrix.
- Rows usually correspond to observations (e.g. samples, replicates, patients), and columns to different features characterizing the observation.

Data in a matrix format

- Datasets can often be organized in a form of a rectangular matrix.
- Rows usually correspond to observations (e.g. samples, replicates, patients), and columns to different features characterizing the observation.
- Each matrix entry corresponds to a measured value of a particular feature in a specific sample.
- Features might have different ranges or even different units

Data in a matrix format

- Datasets can often be organized in a form of a rectangular matrix.
- Rows usually correspond to observations (e.g. samples, replicates, patients), and columns to different features characterizing the observation.
- Each matrix entry corresponds to a measured value of a particular feature in a specific sample.
- Features might have different ranges or even different units



A toy example

- Recall the **mtcars** data.frame used in previous lectures.
- The dataset comes from 1974 *Motor Trend* US magazine and comprises multiple aspects of automobile design and performance for 32 automobiles.
- The data contains both categorical and continuous variables in various units.

```
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
```

Microbial ecology

- **Microbial species abundances:** matrices of bacterial counts are used in microbial ecology studies.
- Columns represent different bacterial species (or sequence variants) – here identified by numerical tags.
- The rows are samples in which bacterial counts were measured.
- The (integer) numbers represent the number of times of each of the bacterial species being observed in each of the samples.

```
data("GlobalPatterns", package = "phyloseq")
GPOTUs = as.matrix(t(phyloseq::otu_table(GlobalPatterns)))
GPOTUs[1:4, 6:13]
```

```
## OTU Table:          [4 taxa and 8 samples]
##                      taxa are rows
##         246140 143239 244960 255340 144887 141782 215972 31759
## CL3          0      7      0    153      3      9      0     0
## CC1          0      1      0    194      5     35      3     1
## SV1          0      0      0      0      0      0      0     0
## M31Fcsw      0      0      0      0      0      0      0     0
```

Diabetes clinical dataset

- This dataset measures glucose levels in the blood after fasting (`glufast`), after a test condition (`glutest`) as well as steady state plasma glucose (`steady`) and steady state insulin (`insulin`) for diabetes.
- The sixth variable is a categorical variable indicating group assignment diabetes.

```
diabetes=read.table('data/diabetes.txt', header=TRUE, row.names=1)
diabetes[1:4,]
```

```
##   relwt glufast glutest steady insulin Group
## 1  0.81      80     356    124      55     3
## 3  0.94     105     319    143     105     3
## 5  1.00      90     323    240     143     3
## 7  0.91     100     350    221     119     3
```

Mass spectroscopy

- Columns correspond to mass spectroscopy peaks or molecules identified by their m/z-ratios.
- Rows (samples) correspond to samples taken from either knockout or wild-type mice.
- Matrix entries are measured continuous intensities.

```
metab = t(as.matrix(read.csv("data/metabolites.csv", row.names = NULL)))
metab[1:4, 1:4]
```

```
##                 [,1]         [,2]       [,3]       [,4]
## mz           146.0985    148.7053   310.1505   132.4513
## KOGCHUM1  29932.3582  17055.7001  1132.8198   785.5129
## KOGCHUM2  94067.6112  74631.6910 28240.8510  5232.0499
## KOGCHUM3 146411.3271 147788.7081 64950.4884 10283.0037
```

Expression data

Cell types microarray: the rows are samples from different subjects and different T-cell types and the columns are expressed genes.

	X3968	X14831	X13492	X5108	X16348	X585
HEA26_EFFE_1	-2.61	-1.19	-0.06	-0.15	0.52	-0.02
HEA26_MEM_1	-2.26	-0.47	0.28	0.54	-0.37	0.11
HEA26_NAI_1	-0.27	0.82	0.81	0.72	-0.9	0.75
MEL36_EFFE_1	-2.24	-1.08	-0.24	-0.18	0.64	0.01
MEL36_MEM_1	-2.68	-0.15	0.25	0.95	-0.2	0.17

continuous

mRNA reads: RNA-Seq transcriptome data report the number of sequence reads matching each gene in each of several biological samples.

	FBgn0000017	FBgn0000018	FBgn0000022	FBgn0000024	FBgn0000028	FBgn0000032
untreated1	4664	583	0	10	0	1446
untreated2	8714	761	1	11	1	1713
untreated4	3150	310	0	3	0	672
treated1	6205	722	0	10	0	1698
treated3	3334	308	0	5	1	757

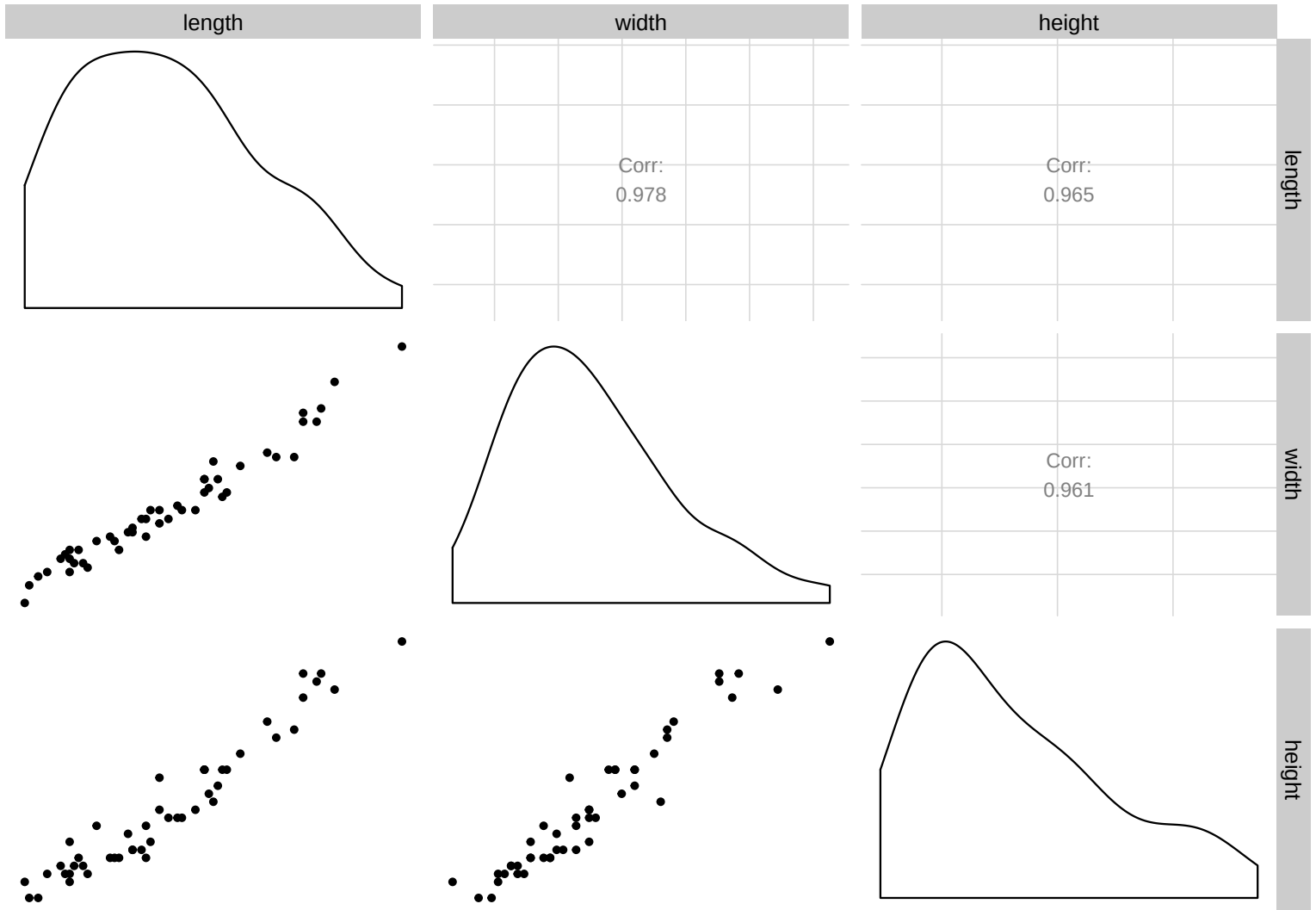
integers

Data summaries and visualization

It is always beneficial to start a multidimensional analysis by checking the simple **one dimensional and two dimensional summary statistics**. We can visualize these using a graphics package that builds on **ggplot2** called **GGally**.

```
library("ggplot2")
library("dplyr")
library("GGally")
turtles = read.table("data/PaintedTurtles.txt", header = TRUE)
ggpairs(turtles[, -1], axisLabels = "none")
```

Painted turtles data



ggally::ggpairs()

```
my_mtcars <- select(mtcars, mpg, wt, qsec, cyl, gear, am) %>%
  mutate(cyl=factor(cyl), gear=factor(gear), am=factor(am))
ggpairs(my_mtcars, aes(color=cyl))
```

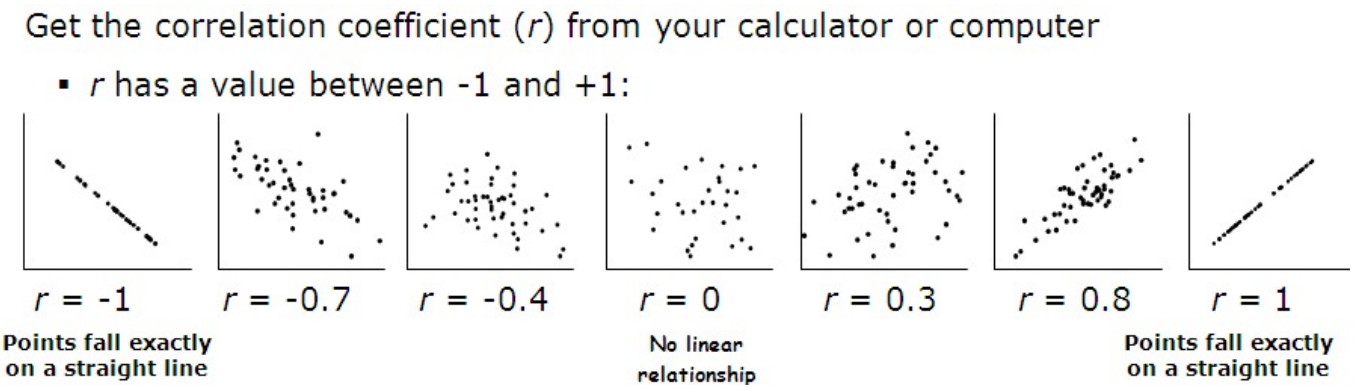
Data summaries

- If we are studying only one variable e.g. just one column of our matrix, we call it one dimensional data.
- A one dimensional data summary could be:
  - a **histogram** that shows that variable's distribution,
  - or we could compute its mean or median — zeroth dimensional summaries of one dimension data.

Correlation coefficients

- When considering two variables x and y measured together on a set of observations, the correlation coefficient measures how the variables co-vary linearly.
- This is a single number summarizing two dimensional data, its formula involves the sample mean of x and y.

$$\hat{\rho} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$



Data preprocessing

Centering

- We usually center the cloud of points around the origin; the most common way of doing this is to make new variables whose means are all zero.

Scaling

- In many cases, different variables are measured in different units, and at different scales (usually measured with variance).
- For instance, `mpg` vary between 10 and 34, miles per gallon and car weight between 1.5k and 5.4k lbs. It would be hard to compare in their original form.

Centering and scaling in R

```
my_mtcars <- select(mtcars, mpg, wt, qsec, hp)
apply(my_mtcars, 2, mean)
```

```
##       mpg        wt      qsec        hp
##  20.09062   3.21725  17.84875 146.68750
```

```
apply(my_mtcars, 2, sd)
```

```
##        mpg         wt       qsec         hp
##  6.0269481  0.9784574  1.7869432 68.5628685
```

```
my_mtcars_centered <- scale(my_mtcars, center=TRUE, scale=FALSE)
apply(my_mtcars_centered, 2, mean)
```

```
##          mpg           wt         qsec           hp
## 4.440892e-16 3.469447e-17 9.436896e-16 0.000000e+00
```

```
apply(my_mtcars_centered, 2, sd)
```

```
##        mpg         wt       qsec         hp
##  6.0269481  0.9784574  1.7869432 68.5628685
```

Centering and scaling in R

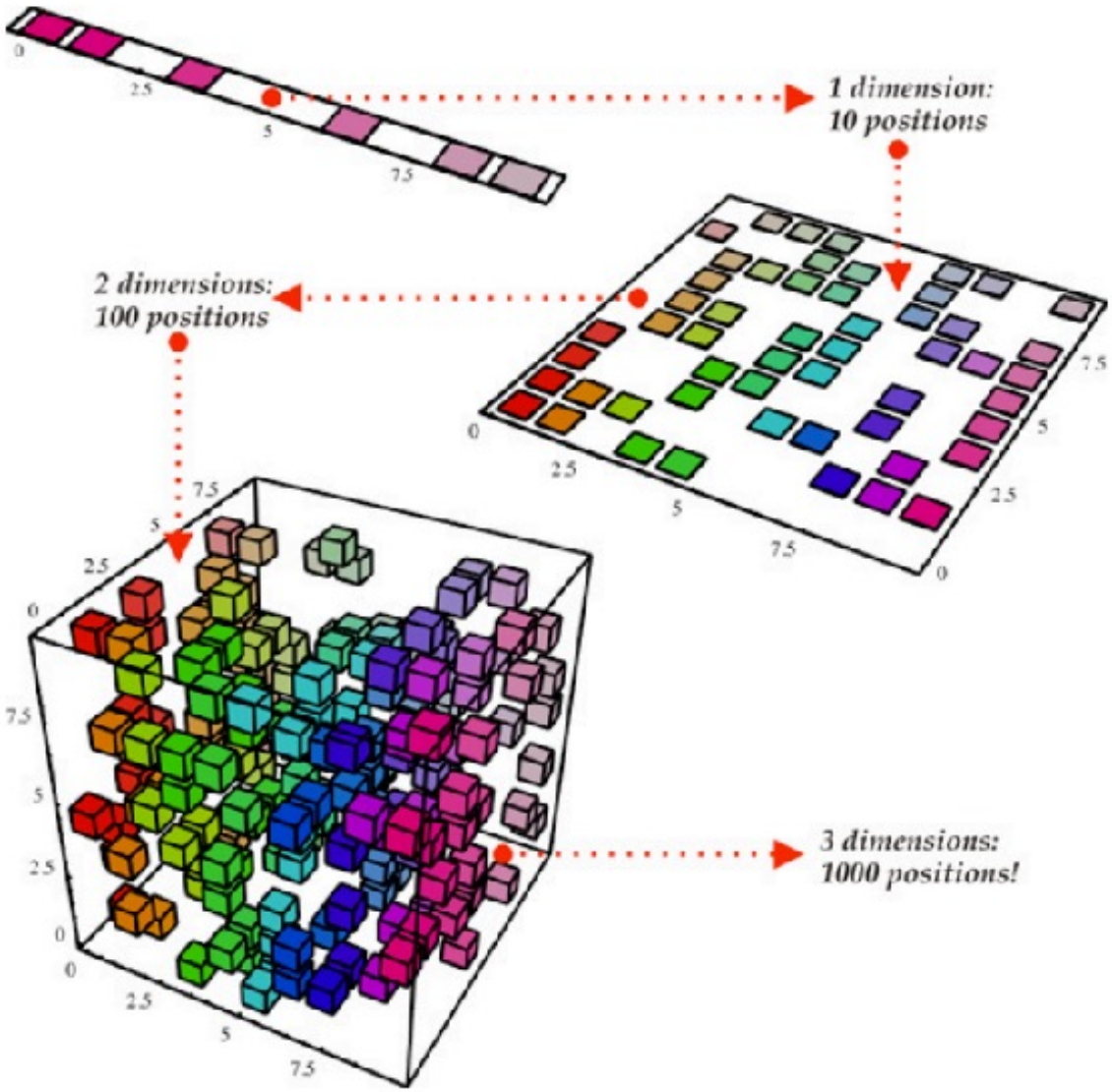
```
my_mtcars_centered_scaled <- scale(my_mtcars)
apply(my_mtcars_centered_scaled, 2, mean)
```

```
##          mpg           wt         qsec           hp
## 7.112366e-17 4.681043e-17 5.299580e-16 1.040834e-17
```

```
apply(my_mtcars_centered_scaled, 2, sd)
```

```
##  mpg   wt qsec   hp
##    1    1    1    1
```

Dimensionality reduction



Why do you need to reduce dimensions?

- Most of real-life datasets are now **high dimensional**
  e.g. genetic sequencing data, medical records data, user internet activity data, etc.

Why do you need to reduce dimensions?

- Most of real-life datasets are now **high dimensional**
  e.g. genetic sequencing data, medical records data, user internet activity data, etc.
- Dimensionality reduction can serve as a feature extraction method which reduce the number of variables without the loss of signal.

Why do you need to reduce dimensions?

- Most of real-life datasets are now **high dimensional**
  e.g. genetic sequencing data, medical records data, user internet activity data, etc.
- Dimensionality reduction can serve as a feature extraction method which reduce the number of variables with little loss of signal.
- The methods can be used to:
  - compress the data,
  - remove redundant features and noise
  - increase accuracy of learning methods by avoiding overfitting and the curse of dimensionality.

Lower-dimensional projections

- We will use geometrical projections that take points in higher dimensional spaces and **project them down onto lower dimensions**.
- In particular, we will explain in detail one technique called **principal component analysis**.
- PCA is primarily an **exploratory** technique that produces maps that show the relations between variables and between observations in a useful way.

History of PCA

- PCA was invented in 1901 by Karl Pearson as a way to reduce a two-variable scatterplot to a single coordinate.
- It was again independently developed by Harold Hotelling in the 1930s. Statisticians in that period used it to summarize a battery of psychological tests run on the same subjects, by constructing overall scores that summarize many variables at once.

History of PCA

Harold Hotelling

Karl Pearson

Good projections

What is this?

Good projections

What is this?

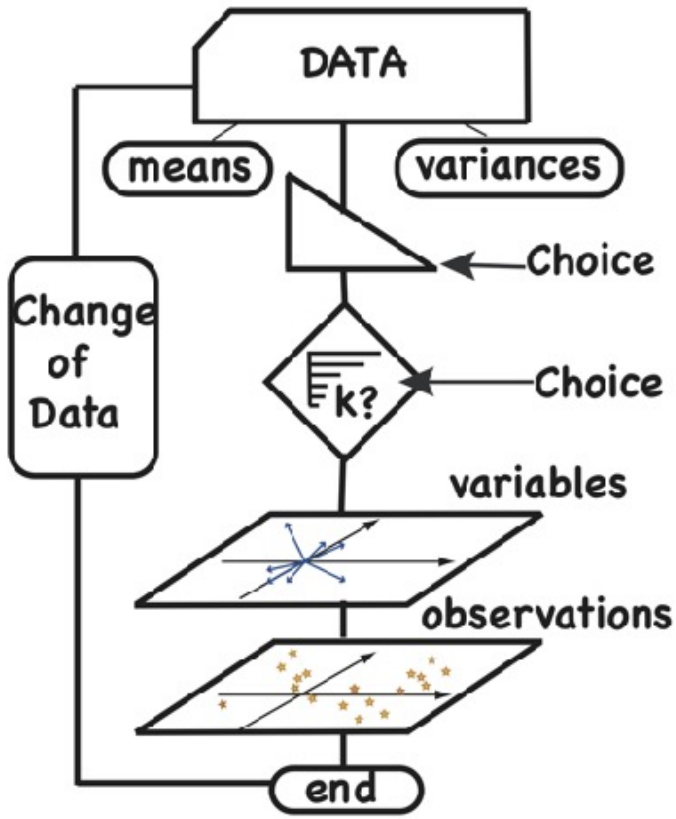
- Which projection do you think is better?
- It's the projection that maximizes the area of the shadow and an equivalent measurement is the sums of squares of the distances between points in the projection.
- We want to see as much of the variation as possible, that's what PCA does.

Principal component analysis

- PCA is an **unsupervised** learning technique because it treats all variables as having the same status.
- PCA is a **visualization** technique which produces maps of both variables and observations.
- PCA is a **linear** technique meaning both that we look for linear relations between variables and that it is based on functions that are linear in the variables and thus particularly easy to compute.

(Linear funtions: $f(ax + by) = af(x) + bf(y)$.)

Principal component analysis



Principal component analysis

- In the diagram, we can see an important step is the choice of k —– the number of components relevant to the data.
- The number of components k is also the rank of the data approximation matrix we chose.
- It is common to set $k = 2$ for the sake of visualization.

What is the maximum number of PCs?

- Suppose, we have a data matrix X with n samples and p variables: $X \in \mathbb{R}^{n \times p}$.
- The maximum possible number of principal components is less than or equal to the minimum of the number of samples and the number of original variables.

$$K \leq \min(n, p).$$

- Suppose we have 5 samples with 23,000 genes measured on them. The maximum number of PCs is 5.

Scree plot

- Usually, not all of PCs are informative, the data matrix can be approximately low-rank, and higher PCs correspond to noise.
- Choosing the number of PCs to retain is an important part of a PCA procedure.

Scree plot

- Usually, not all of PCs are informative, the data matrix can be approximately low-rank, and higher PCs correspond to noise.
- Choosing the number of PCs to retain is an important part of a PCA procedure.
- Unlike clustering, the number of components should be chosen after completing the analysis. The choice of k, the number of PCs, for the data projection requires looking at a screeplot giving the magnitude of the eigenvalues.

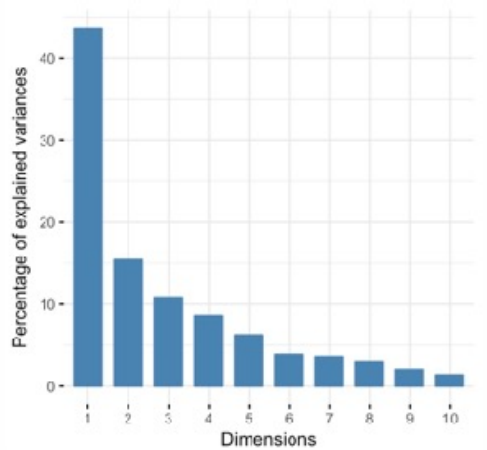


Figure 7.29: Screeplot showing the eigenvalues for the mice data.

Scree plot

- There are situations when the PCs are ill-defined: when two or three successive PCs have very similar variances. Then, (eigen)vectors are not meaningful individually and one cannot interpret their loadings. A very slight change in one observations could give a completely different set of three vectors

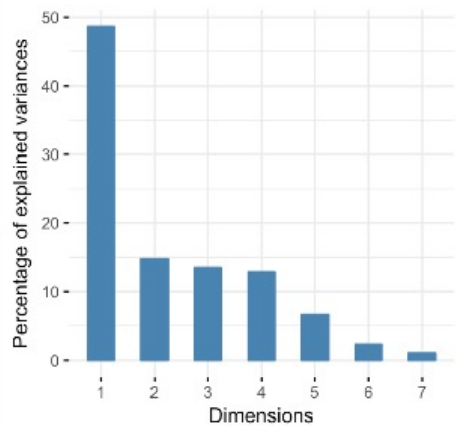
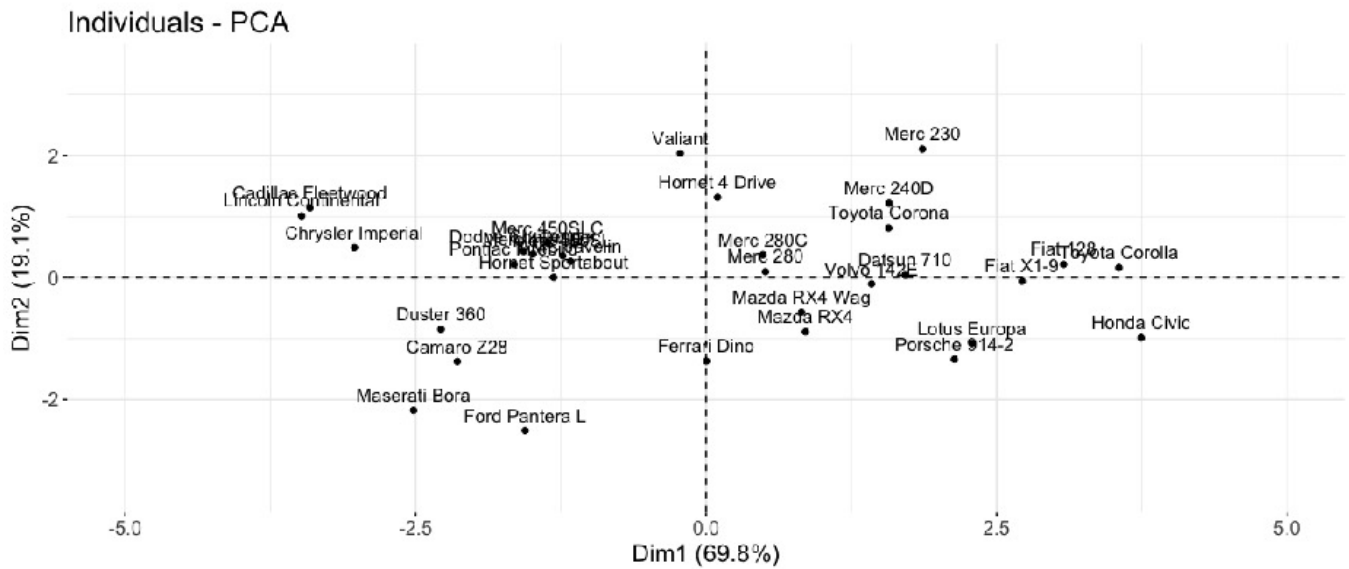


Figure 7.26: A screeplot showing 'dangerously' similar variances. Choosing to cutoff at a hard threshold of 80% of the variance would give unstable PC plots. With so such cutoff, the axes corresponding to the 3D subspace of 3 similar eigenvalues are unstable and cannot be individually interpreted.

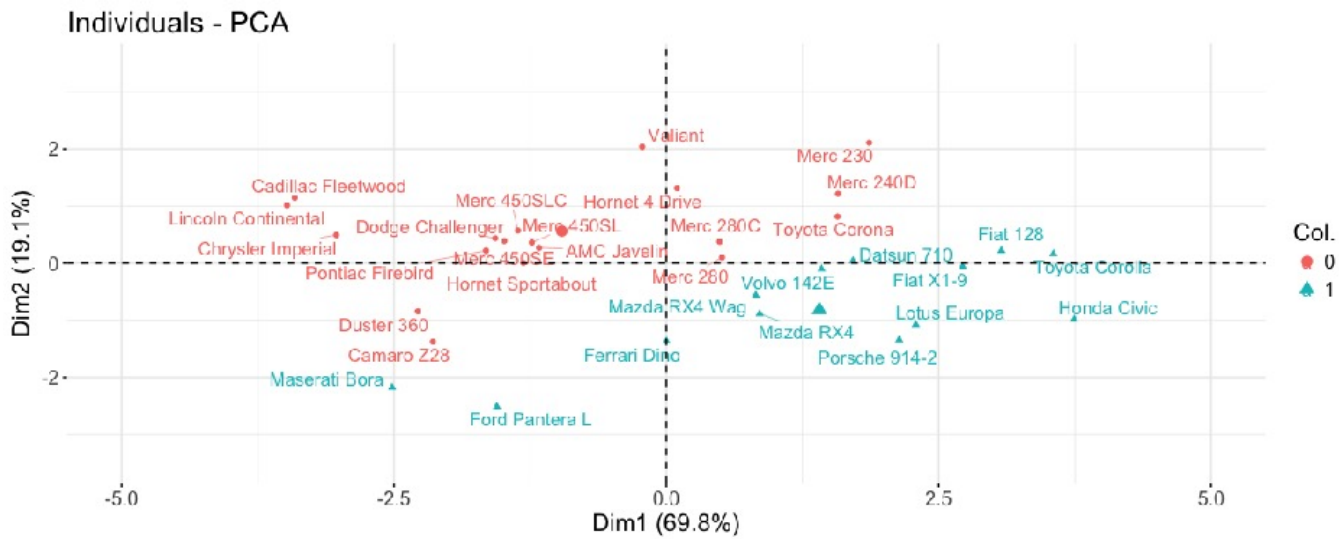
Projection of the samples

- PCA was performed for `mtcars` dataset (only six continuous variables).
- Recall that the observations in the dataset are cars (from 70s so they might not sound familiar).
- Note that the first PC explains already ~70% of the total variance.



Projection of the samples

- We can color the datapoint by additional information available, e.g. whether the car has automatic or manual transmission:



Eigenvalues and aspect ratio

- Remember that we said the variance of PCs are given by the corresponding eigenvalues.

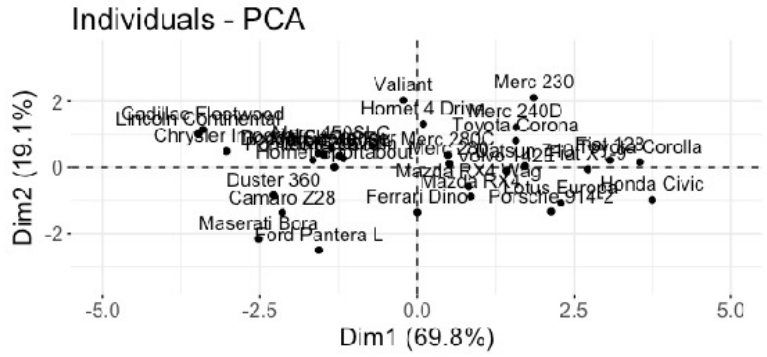
Eigenvalues and aspect ratio

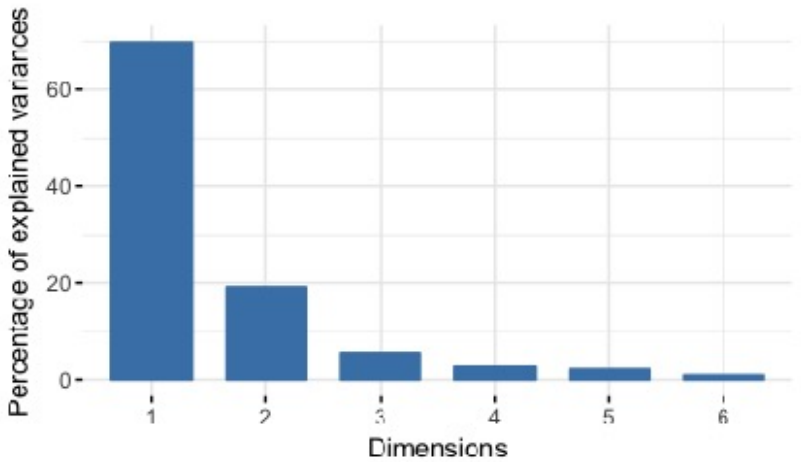
- Remember that we said the variance of PCs are given by the corresponding eigenvalues.
- Relative eigenvalues (divided by the total sum of all eigenvalues) give the fraction of variance explained.

Eigenvalues and aspect ratio

- Remember that we said the variance of PCs are given by the corresponding eigenvalues.
- Relative eigenvalues (divided by the total sum of all eigenvalues) give the fraction of variance explained.
- Thus, the aspect ratio of the PCA samples projection map should reflect the ratio between the square root of eigenvalues.

Eigenvalues and aspect ratio



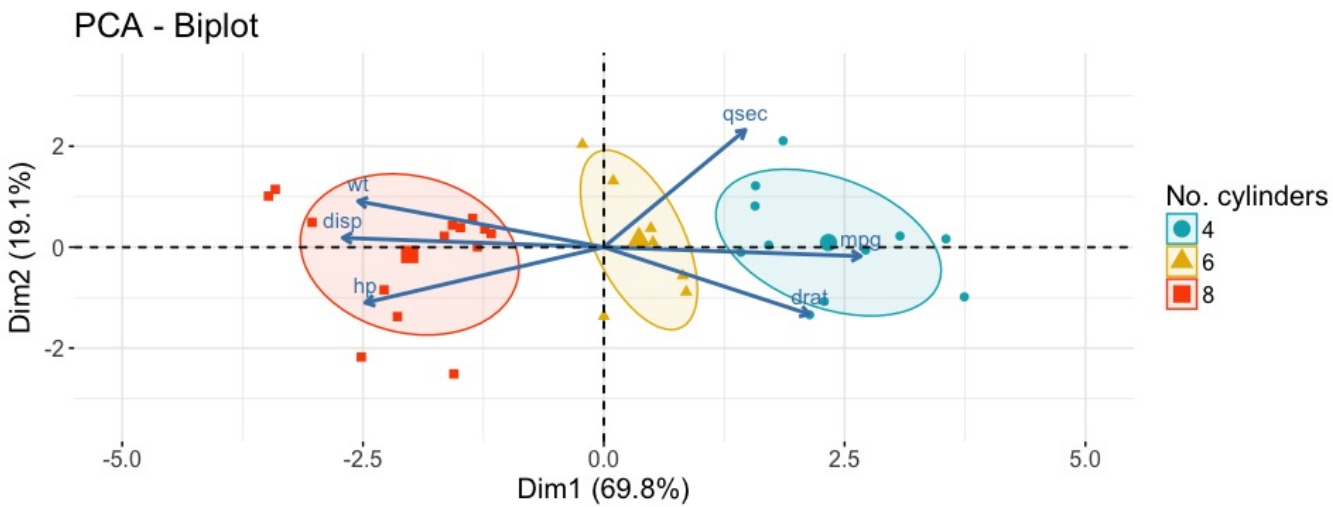


PCA biplot

- It is often useful to have both the variables and the samples on the same map.
- This simultaneous representation of both the observation and samples is called a biplot.
- Note that, the biplot is only useful when there is a small number of variables and observations in the data set; otherwise the final plot would be overcrowded.
- The coordinate of individuals and variables are not constructed on the same space. Therefore, in the biplot, you should mainly focus on the direction of variables but not on their absolute positions.

PCA biplot

- Roughly speaking, a biplot can be interpreted as follows: a sample that is on the same side of a given variable has a high value for this variable, whereas a sample that is on the opposite side of a given variable has a low value for this variable.
- In this example, we can see that the cars with eight cylinders tend to have hight horsepower and weigh a lot, and they also have low mileage per hour. Cars with four cylinders are the exact opposite, which makes sense!



Running PCA in R

Wine dataset

- This dataset contains chemical measurements on different wines.
- We also have information about the class of wine each one belongs to, but we will not use it for any computations, just for results interpretation after completing the analysis.
- If you ever want to hear a dinner table-appropriate explanation of PCA, I recommend looking up this stats-stackexchange post which is also about wines!

Wine dataset

	Alcohol	MalicAcid	Ash	AlcAsh	Mg	Phenols	Flav	NonFlav Phenols	Proa	Color	Hue	OD	Proline
1	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065
2	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050
3	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185
4	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480
5	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735
6	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450

Center and scale the Data

```
load("data/wine.RData")
apply(wine, 2, mean)
```

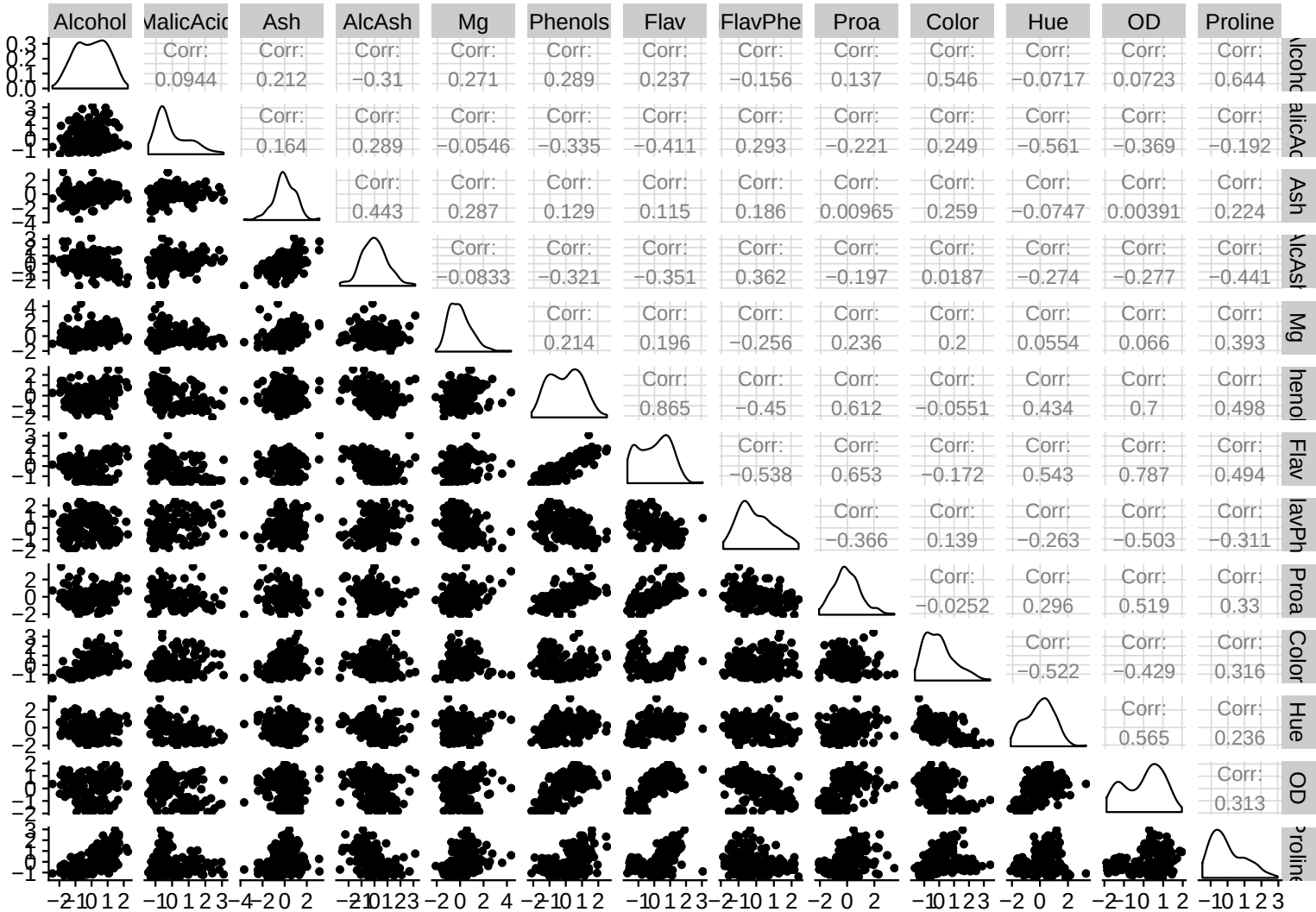
```
##        Alcohol      MalicAcid            Ash         AlcAsh             Mg
##     13.0006180      2.3363483      2.3665169     19.4949438     99.7415730
##        Phenols           Flav NonFlavPhenols           Proa          Color
##      2.2951124      2.0292697      0.3618539      1.5908989      5.0580899
##            Hue             OD        Proline
##      0.9574494      2.6116854    746.8932584
```

```
apply(wine, 2, sd)
```

```
##        Alcohol      MalicAcid            Ash         AlcAsh             Mg
##      0.8118265      1.1171461      0.2743440      3.3395638     14.2824835
##        Phenols           Flav NonFlavPhenols           Proa          Color
##      0.6258510      0.9988587      0.1244533      0.5723589      2.3182859
##            Hue             OD        Proline
##      0.2285716      0.7099904    314.9074743
```

```
wine_scaled = scale(wine)
```

2D summaries



Use R functions to compute PCA

- To compute PCA in R, you can use any of the following functions: `princomp`, `prcomp`, `ade4::dudi.pca` or even `svd`.
- Here we will use `dudi.pca` from `ade4` package. For others see the textbook.
- Note that, there is no need to center and scale the data before calling `dudi.pca`.

```r
library(ade4)
winePCA = dudi.pca(wine, nf=5, scale=TRUE, center=TRUE, scannf=FALSE)
class(winePCA)
```

```
## [1] "pca"  "dudi"
```

- Note that we set nf $= 5$, which means we only retain 5 PCs. The rest will not be included in the result.
- The output of `dudi.pca()` is of class both "pca" and "dudi", but is basically a list containing many elements.
- `scan = FALSE` surpasses automatic printing of the the screeplot.

Elements of the output

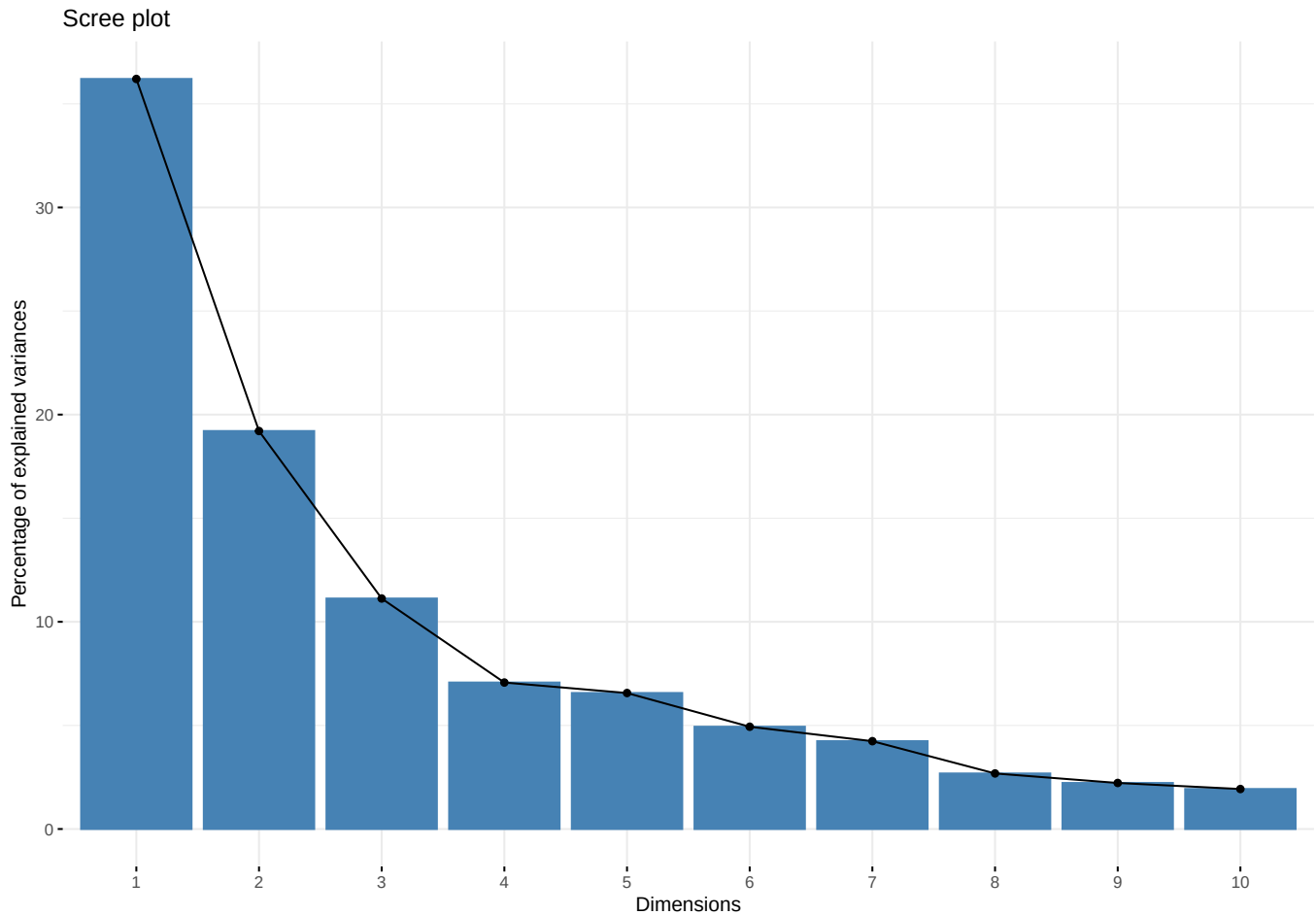
```
names(winePCA)
```

```
##  [1] "tab"  "cw"   "lw"   "eig"  "rank" "nf"   "c1"   "li"   "co"   "l1"
## [11] "call" "cent" "norm"
```

tab	the data frame to be analyzed depending of the transformation arguments (center and scale)
cw	the column weights
lw	the row weights
eig	**the eigenvalues**
rank	the rank of the analyzed matrice
nf	the number of kept factors
c1	the column normed scores i.e. the principal axes
l1	the row normed scores
co	**the column coordinates**
li	**the row coordinates i.e. the principal components**
call	the call function
cent	the p vector containing the means for variables (Note that if center = F, the vector contains p 0)
norm	the p vector containing the standard deviations for variables i.e. the root of the sum of squares deviations of the values from their means divided by n (Note that if norm = F, the vector contains p 1)

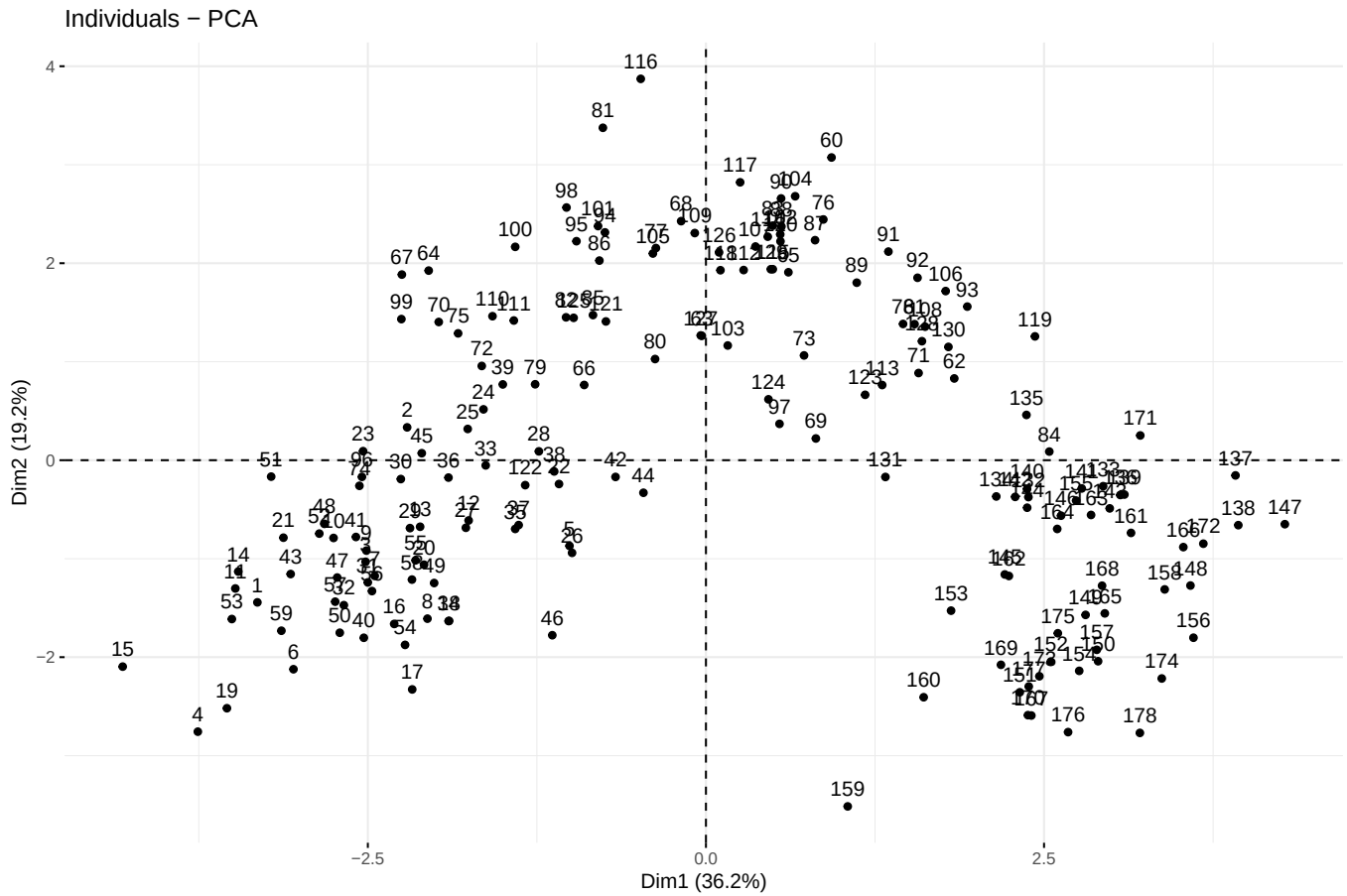
Scree plot

```
library(factoextra)
fviz_eig(winePCA)
```



Sample projection

```
eig_ratio <- winePCA$eig[2]/winePCA$eig[1]
fviz_pca_ind(winePCA) + coord_fixed(sqrt(eig_ratio))
```

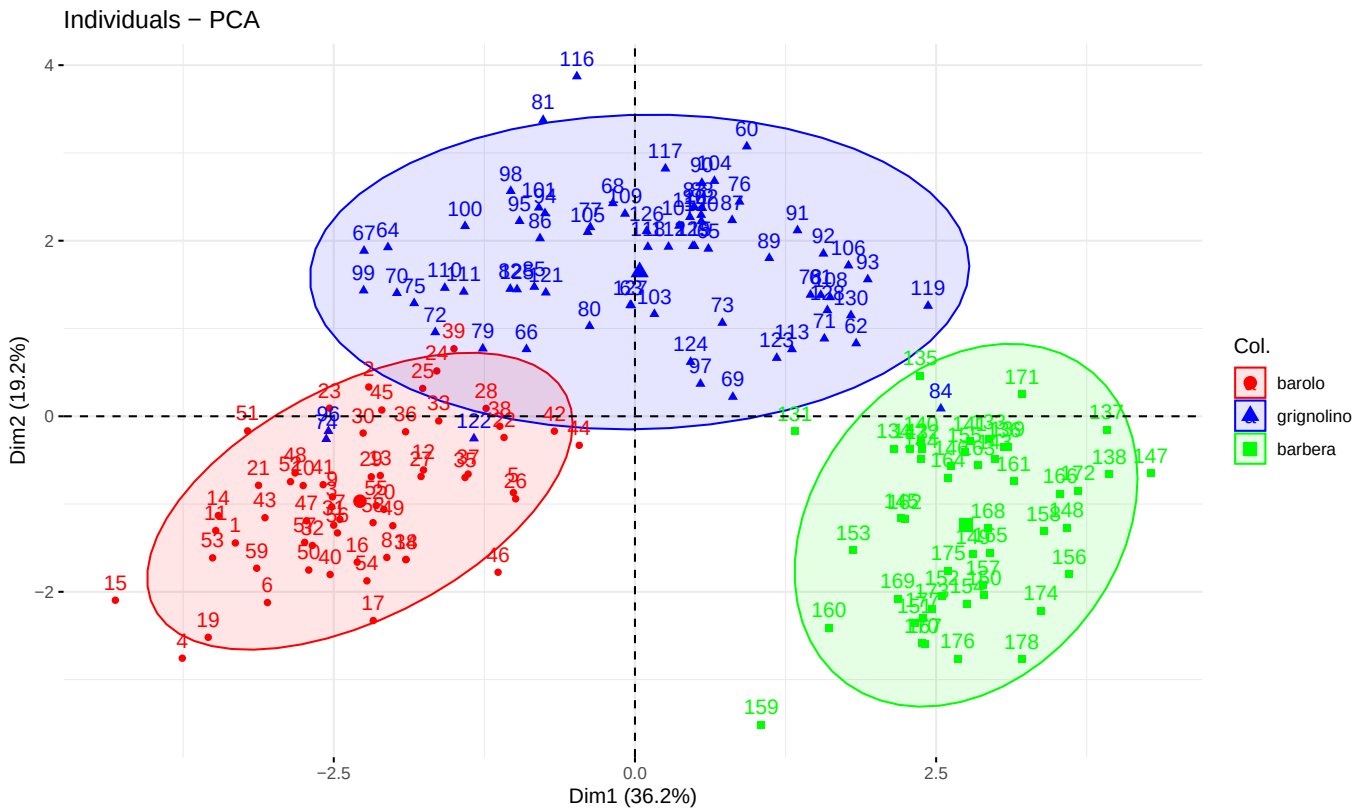


Sample projection with coloring by covariates

```
load("data/wineClass.RData");
# table(wine.class)
fviz_pca_ind(winePCA, col.ind=wine.class, palette=c('red','blue','green')) +
  coord_fixed(sqrt(eig_ratio))
```

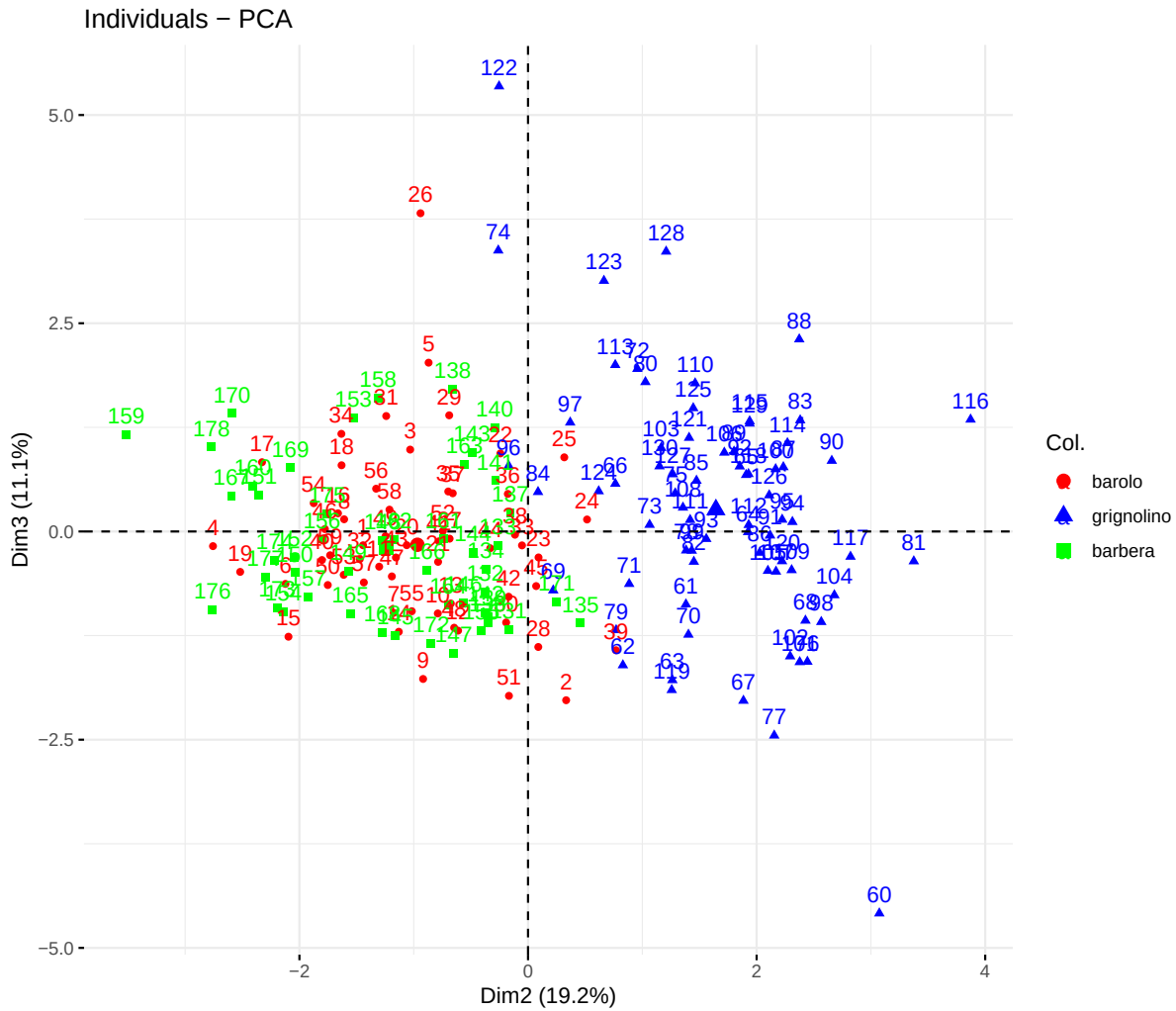
Including ellipses

```
fviz_pca_ind(winePCA, col.ind=wine.class, palette=c('red','blue','green'),
      addEllipses = TRUE, ellipse.level = 0.9) + coord_fixed(sqrt(eig_ratio))
```



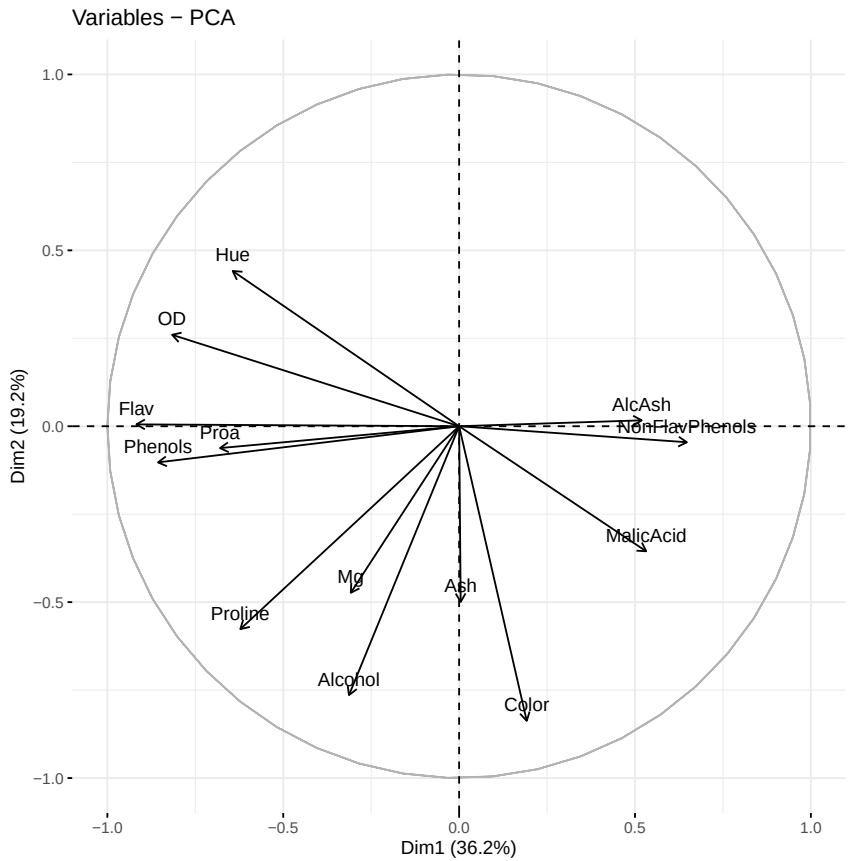
Sample projection on other axes

```
fviz_pca_ind(winePCA,  axes=2:3, col.ind=wine.class,
    palette=c('red','blue','green')) + coord_fixed(sqrt(eig_ratio))
```



Correlation circle for variables

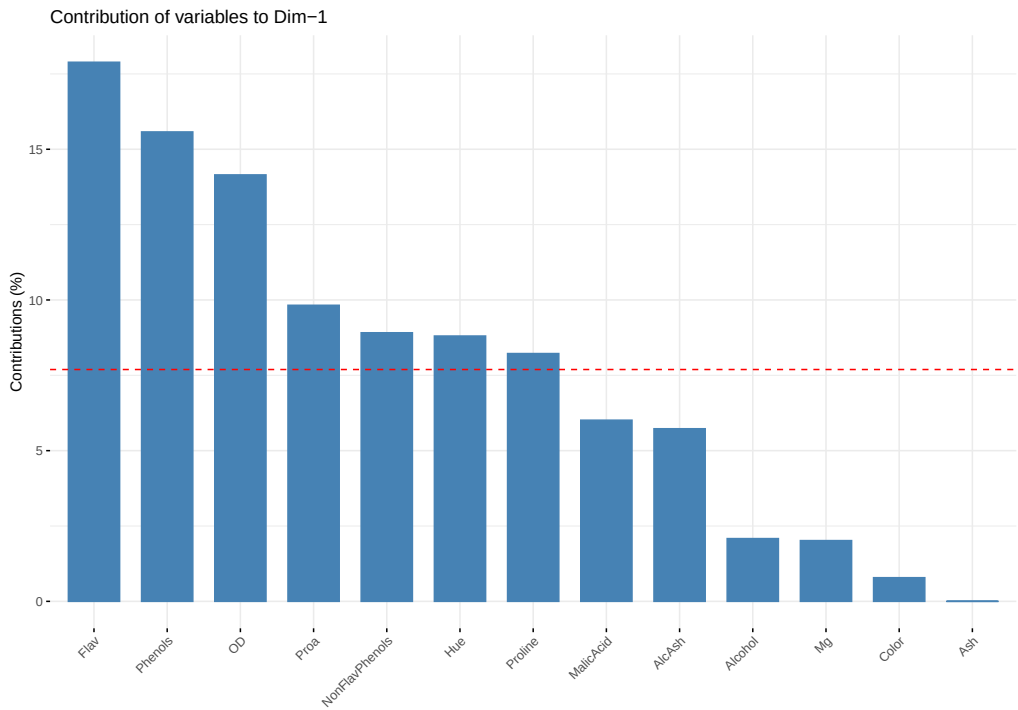
```
fviz_pca_var(winePCA)
```



Variable contribution to PCs

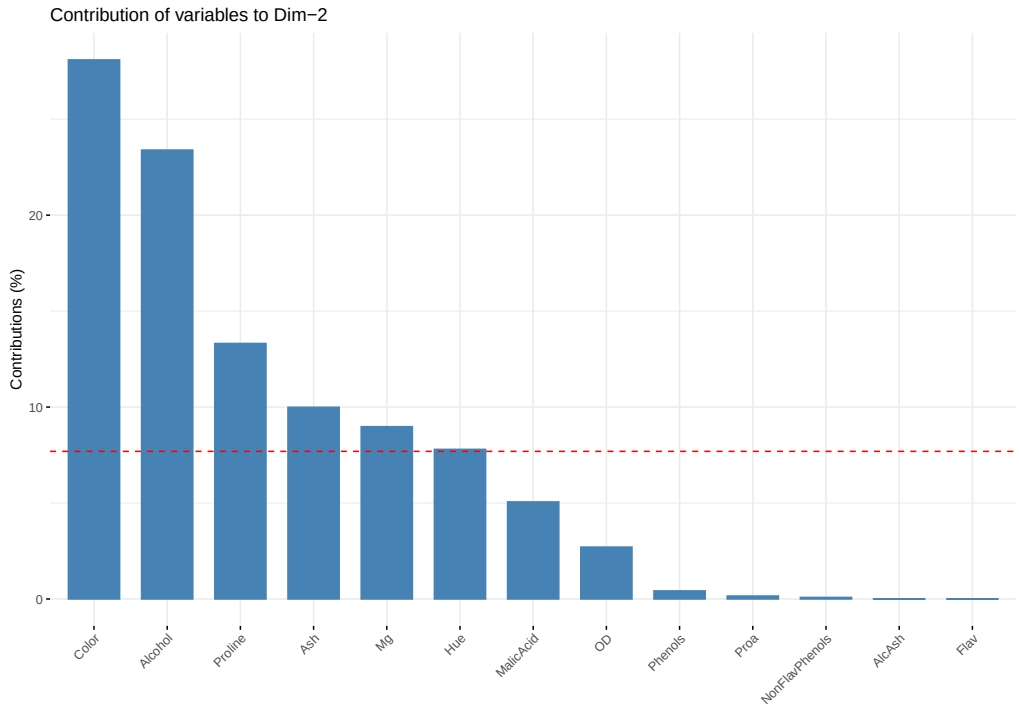
Contribution is the squared correlation of the variable to the dimension divided by the sum of squared correlations for all variables.

```
fviz_contrib(winePCA, choice="var", axes=1)
```



Variable contribution to PCs

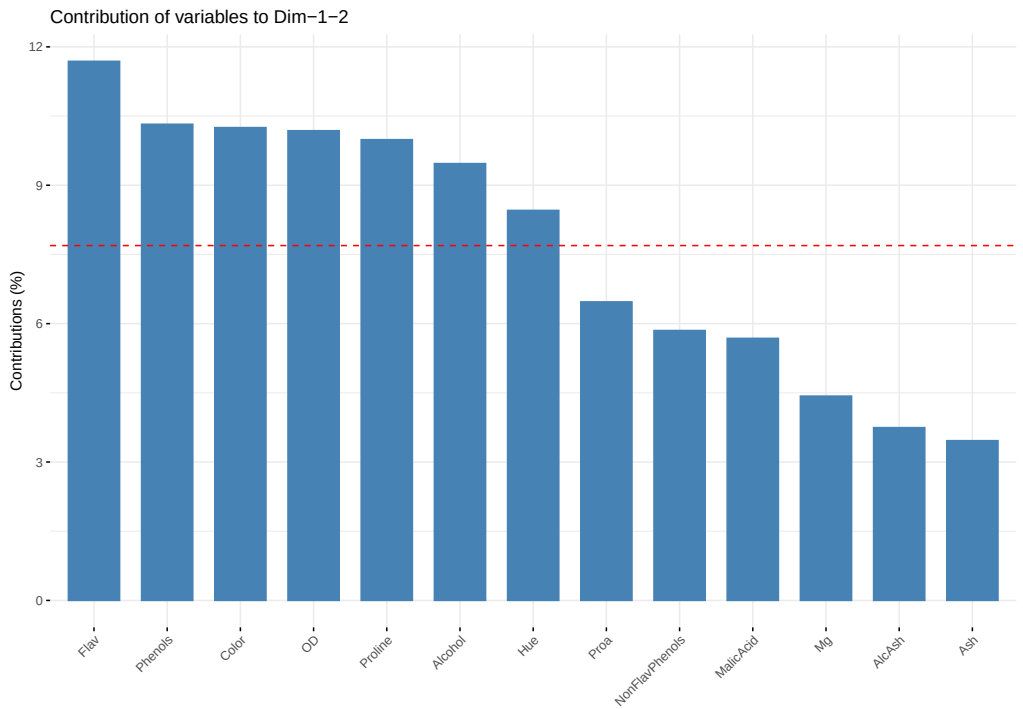
```
fviz_contrib(winePCA, choice="var", axes=2)
```



The red dashed line on the graph above indicates the expected average contribution, as if the contribution of the variables was even.

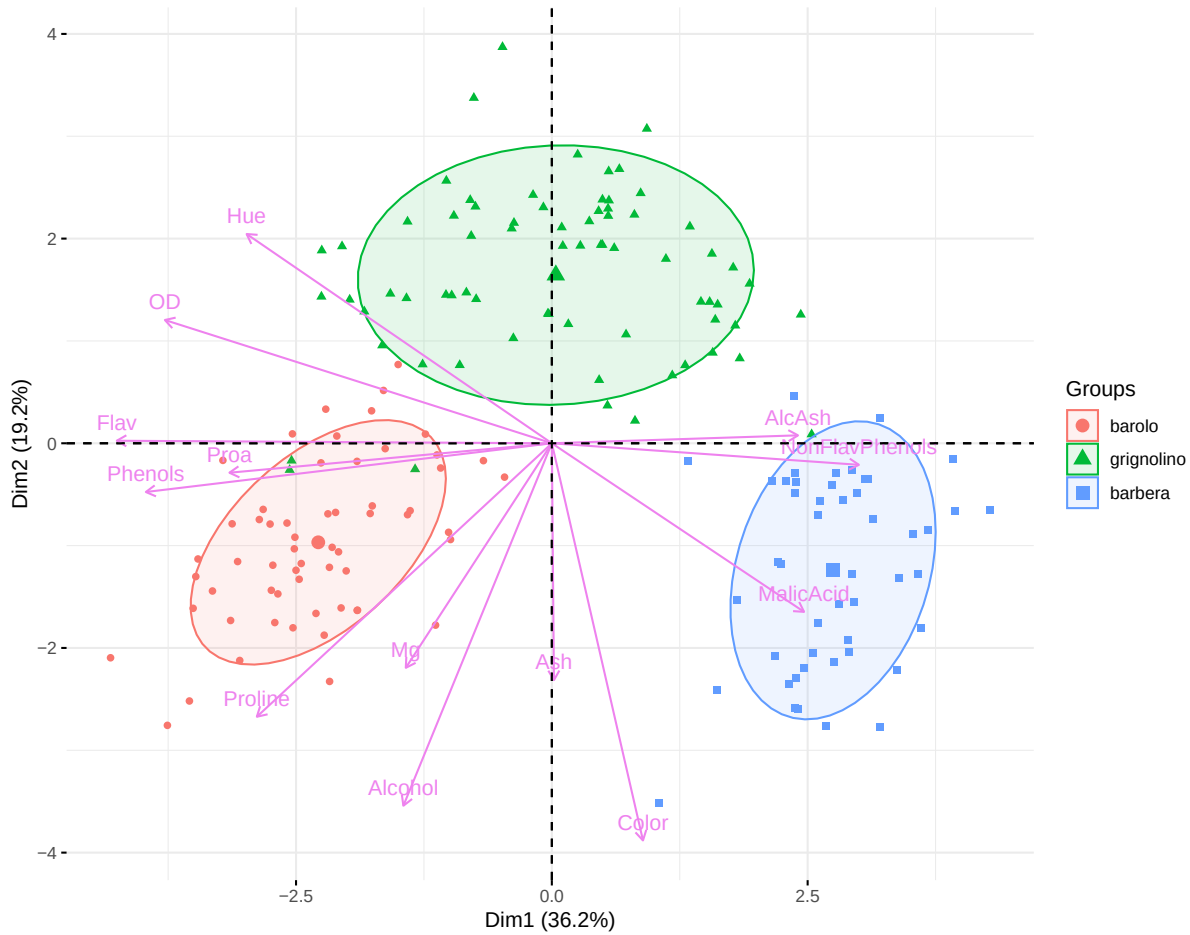
Variable contribution to PCs

```
fviz_contrib(winePCA, choice="var", axes=1:2)
```



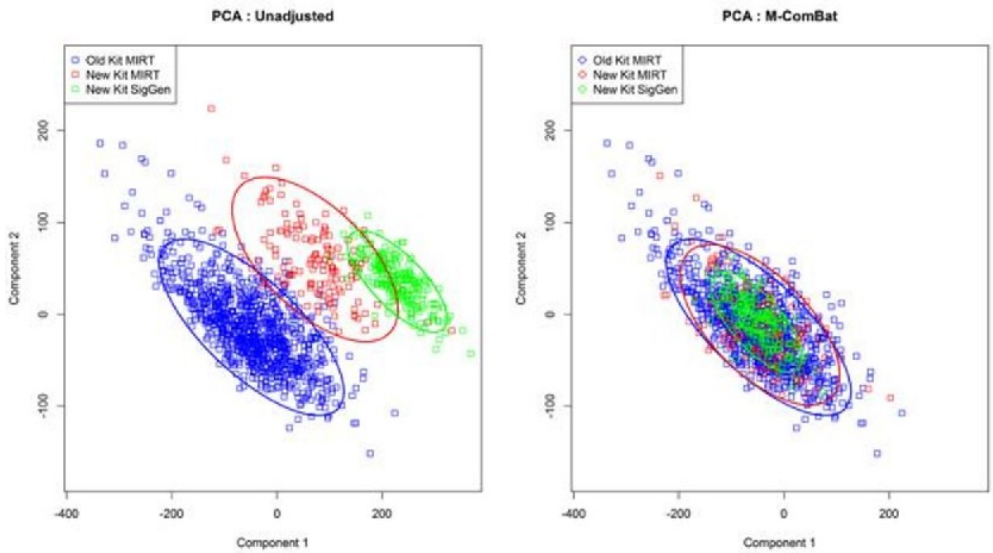
PCA biplot with everything together

```
fviz_pca_biplot(winePCA, geom = "point", habillage = wine.class,
   col.var = "violet", addEllipses = TRUE, ellipse.level = 0.69) +
   ggtitle("") + coord_fixed()
```



Discovering batch effects

- PCA can be used to discover batch effects.
- If there are batch effects, you can use tools such as `ComBat` which are available in `sva` package .



See Stein et al. (2014).