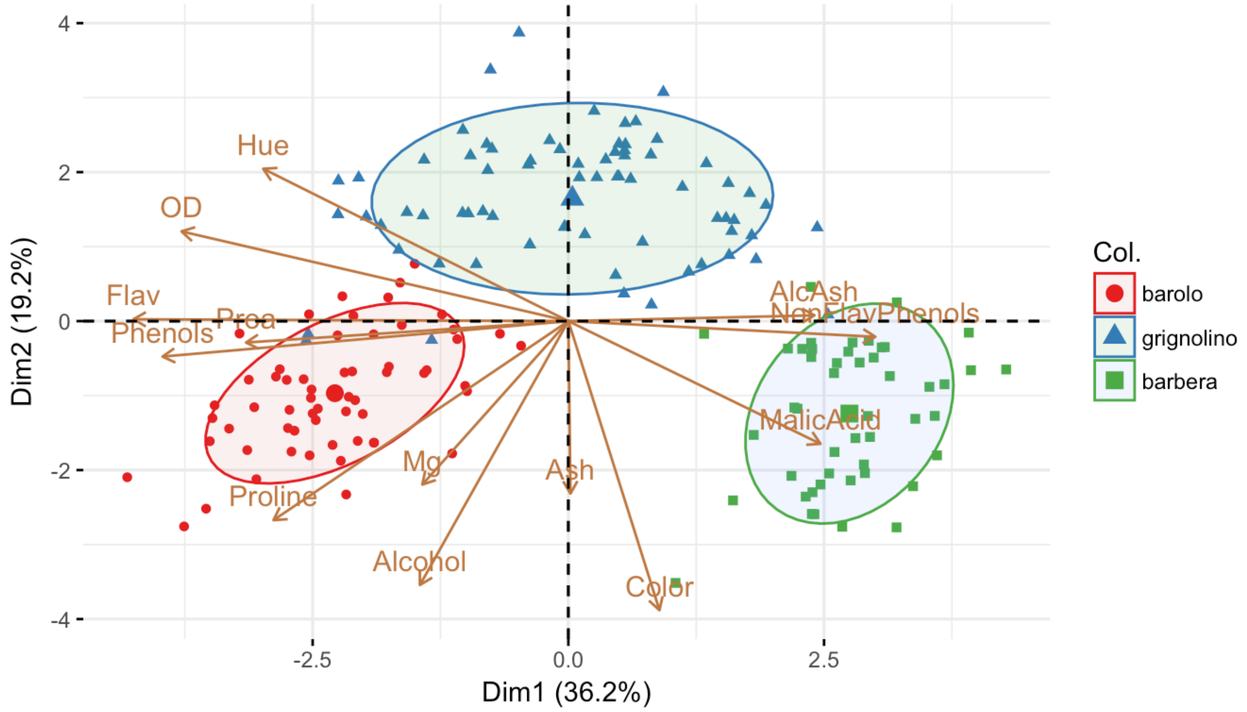


Multivariate Analysis

UBDS3 2024
Susan Holmes
Lan Huong Nguyen
Wolfgang Huber

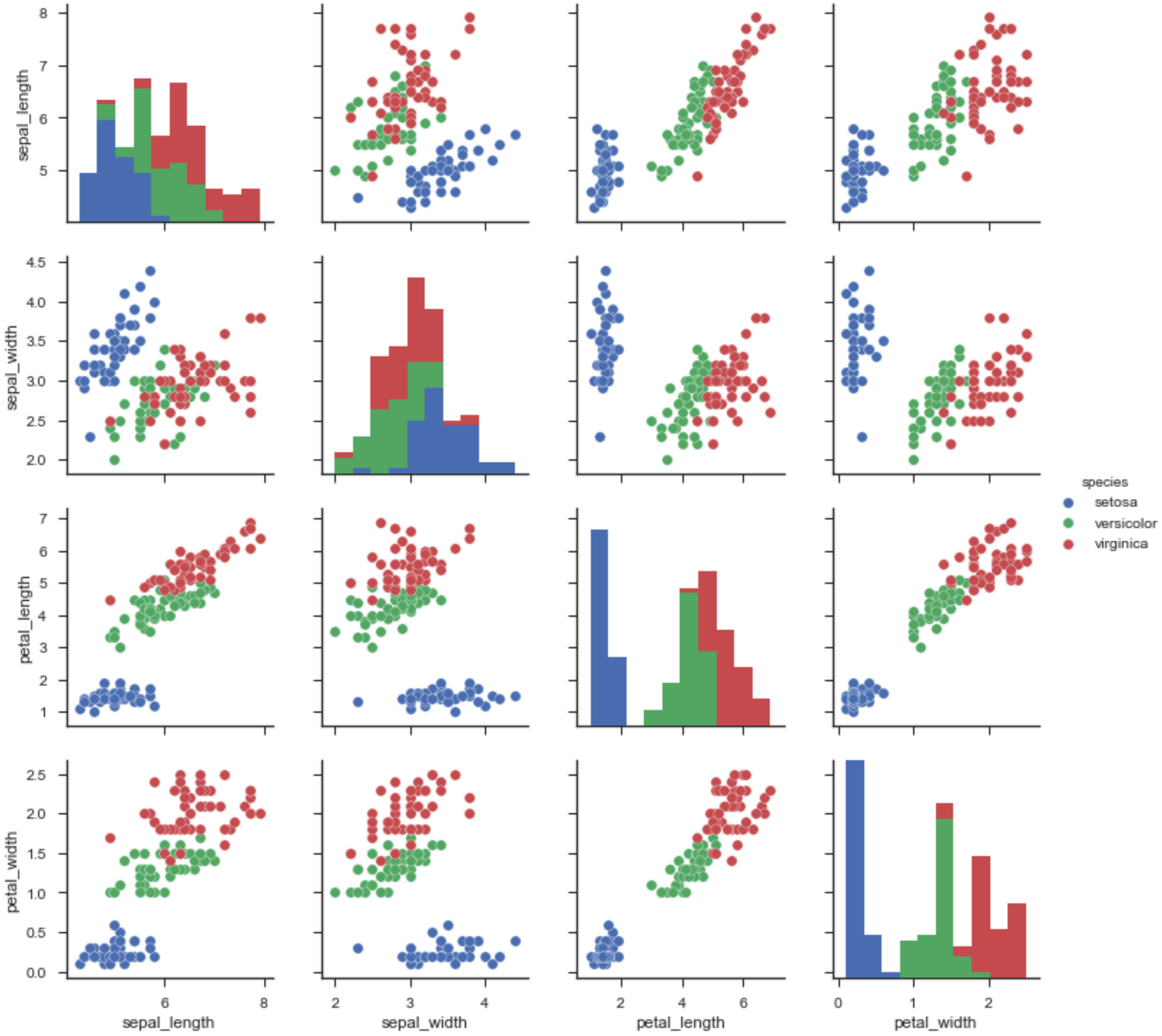
What is multivariate analysis?

- Many studies collect information on **multiple variables** repeated for each observation.
- Modern datasets are often **high-dimensional**, with thousands to millions of variables ("dimensions") per sample.
- Multivariate Analysis is an umbrella term for techniques to analyze more than one variable at a time. It encompasses many statistical methods:
  - PCA,
  - Correspondence Analysis,
  - Factor Analysis,
  - Multidimensional Scaling, t-SNE, UMAP, …
  - Autoencoders

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.



Goals for this lecture

- See **examples of multivariate data** coming up in biological studies and others.
- Learn how to **preprocess, center and rescale data** before the analysis.
- Understand the **importance of correlations** between variables.
- Recognize why **dimensionality reduction** is useful.
- Build new variables, called **principal components** (PC), that are more useful than the original measurements.
- See **what is "under the hood" of PCA** and learn how to choose the number of principal components.
- Generate **helpful visualizations of PCA** results.
- **Run through a complete PCA analysis** from start to finish.

Datasets

Data format

- Datasets can often be organized as a rectangular table.
- Rows correspond to experimental or study units (e.g., environmental sample, patient biopsy, customer order), and
- Columns to variables (a.k.a. features, measurements). Can have different data types, units.
- “Tidy data”: https://r4ds.had.co.nz/tidy-data.html
  - R: data.frame
  - Python: Pandas DataFrame;
  - SQL: table
  - CSV file
  - Excel: sheet (…)
- Special case: matrix—all variables have same data type & units

Motor cars

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

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3,9	2,62	16,46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3,9	2,875	17,02	0	1	4	4
Datsun710	22,8	4	108	93	3,85	2,32	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	3,44	17,02	0	0	3	2
Valiant	18,1	6	225	105	2,76	3,46	20,22	1	0	3	1

Turtles

- Three dimensions of biometric measurements on painted turtles (Jolicoeur, 1960)

```
turtles[1:4,]
```

```
##   sex length width height
## 1   f     98    81     38
## 2   f    103    84     38
## 3   f    103    86     42
## 4   f    105    86     40
```

Athletes (Decathlon)

Athletes' performances in the decathlon

- m100, m400, m1500: performance times in seconds for 100 m, 400 m and 1500 m respectively
- 'm110' is the time taken to finish the 110 m hurdles
- pole is the pole-jump height
- weight is the length in metres the athletes threw the weight.

	m100	long	weight	highj	m400	m110	disc	pole	jave	m1500
1	11.25	7.43	15.48	2.27	48.90	15.13	49.28	4.70	61.32	268.95
2	10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.10	61.76	273.02
3	11.18	7.44	14.20	1.97	48.29	14.81	43.66	5.20	64.16	263.20
4	10.62	7.38	15.02	2.03	49.06	14.72	44.80	4.90	64.04	285.11

Diabetes

- Collected by Reaven and Miller (1979) to study the relationship among blood chemistry measures of glucose tolerance and insulin.
- Glucose levels in the blood after fasting (`glufast`), after a test condition (`glutest`), steady state plasma glucose (`steady`) and steady state insulin (`insulin`) for 145 non-obese adults.
- The last variable is a categorical, indicating diagnostic group membership:
  1=overt diabetic, 2=chemical diabetic, 3=normal

diabetes

##		relwt	glufast	glutest	steady	insulin	Group
##	1	0,81	80	356	124	55	3
##	3	0,94	105	319	143	105	3
##	5	1	90	323	240	143	3
##	7	0,91	100	350	221	119	3

Microbial ecology

- Microbial species abundances are estimated using sequencing.
- Data are aggregated as a matrix of read counts:
  - Columns represent different bacterial species (or seq. variants),
  - Rows correspond to samples that were sequenced,
  - Entries are integers representing the number times a specific bacterial species was observed in each of the samples.

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

`GlobalPatterns` Microbial Abundance Data

##		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

RNA-seq Expression Data

- 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

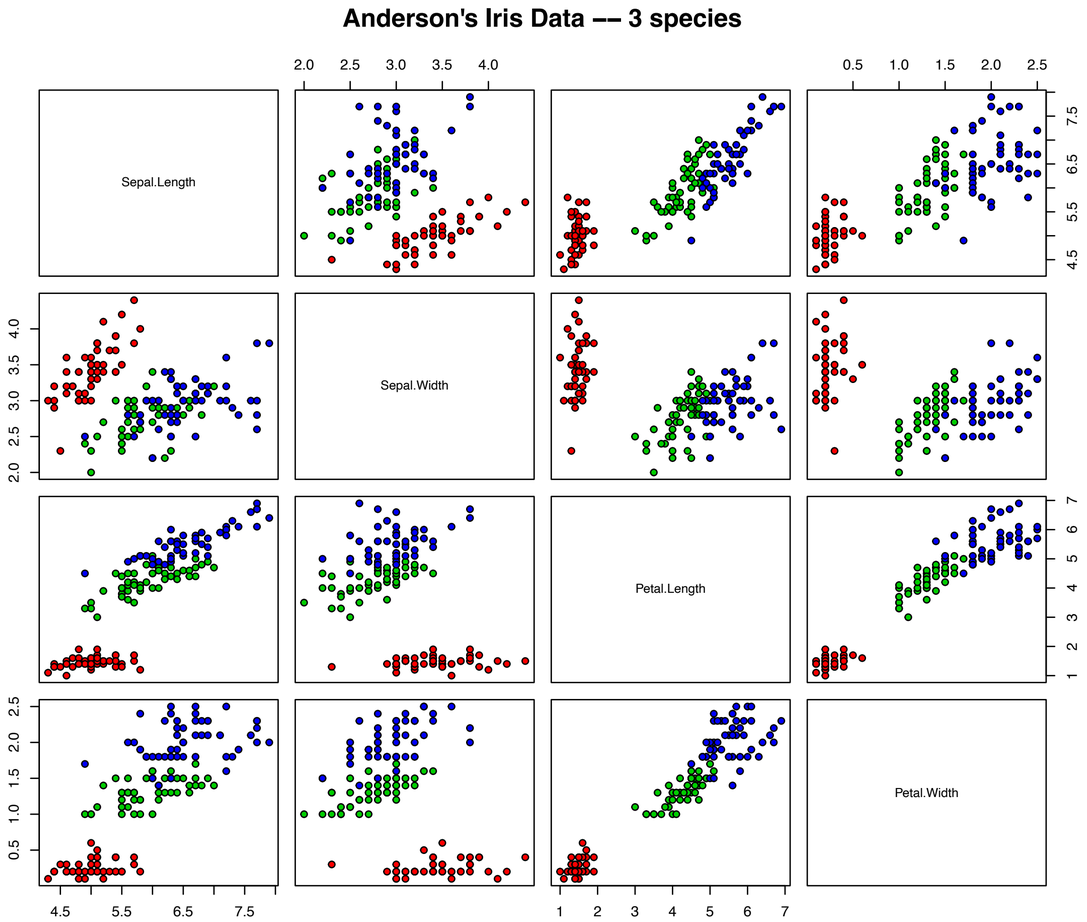
integers

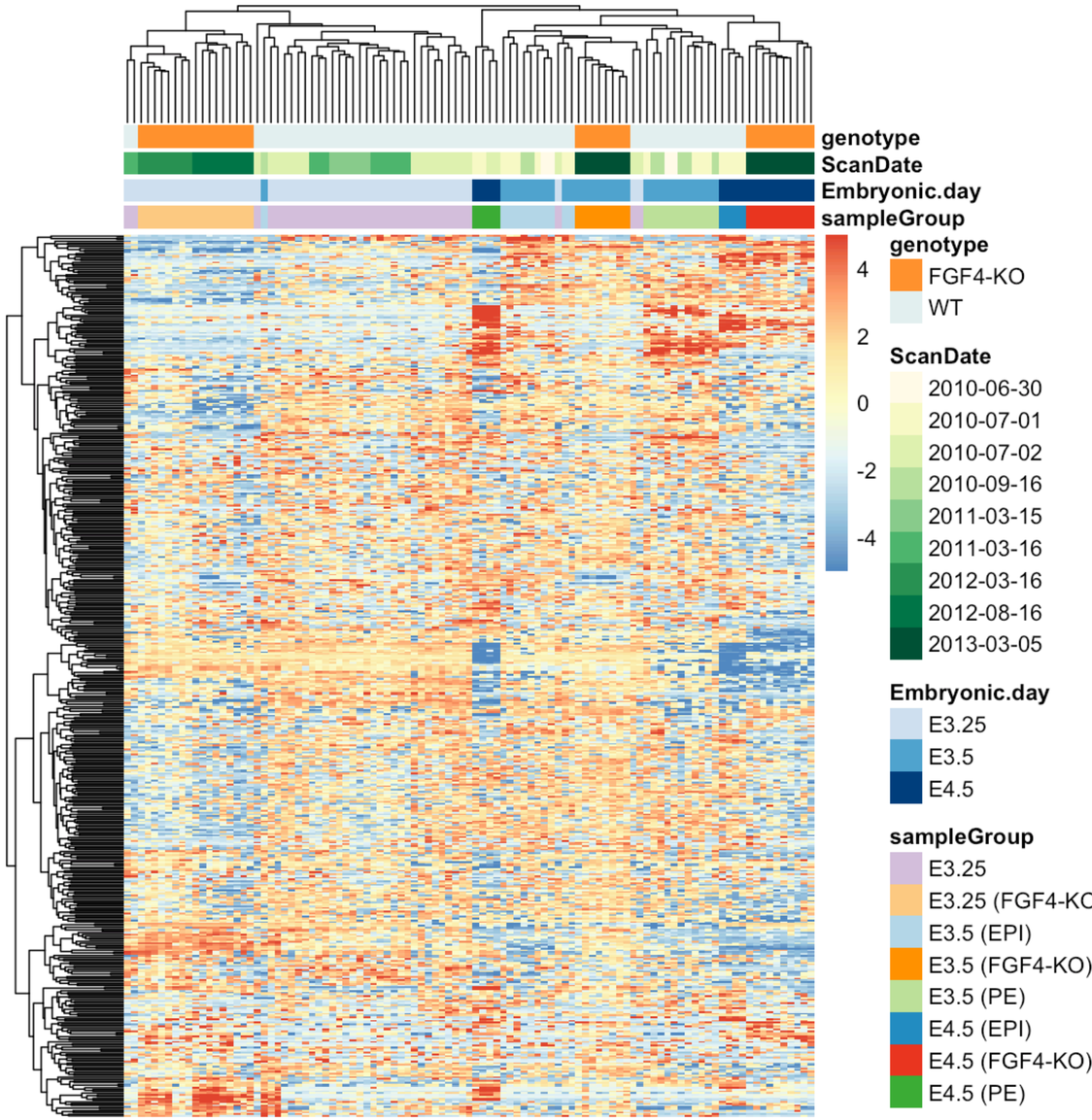
Graphical Exploration

Always start with visual exploration

- Heatmap
- Pairs Plots

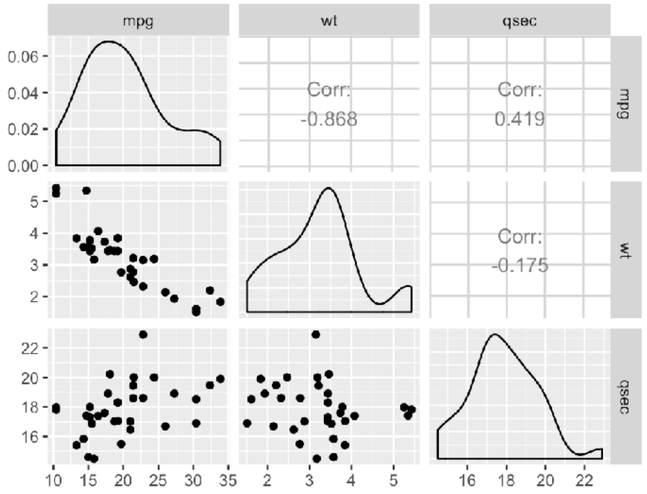
Book chapter https://www.huber.embl.de/msmb/03-chap.html



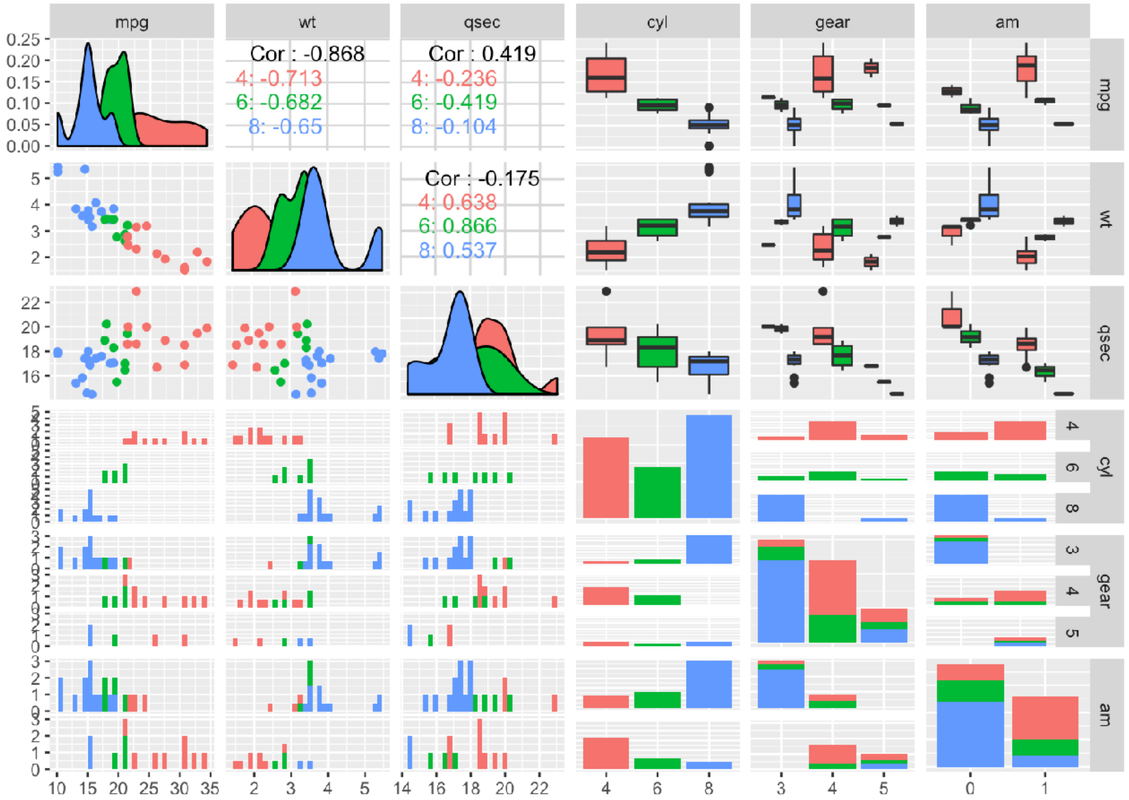


ggally::ggpairs()

```
library(dplyr)
library(GGally)
my_mtcars <- select(mtcars, mpg, wt, qsec)
ggpairs(my_mtcars)
```




```
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))
```



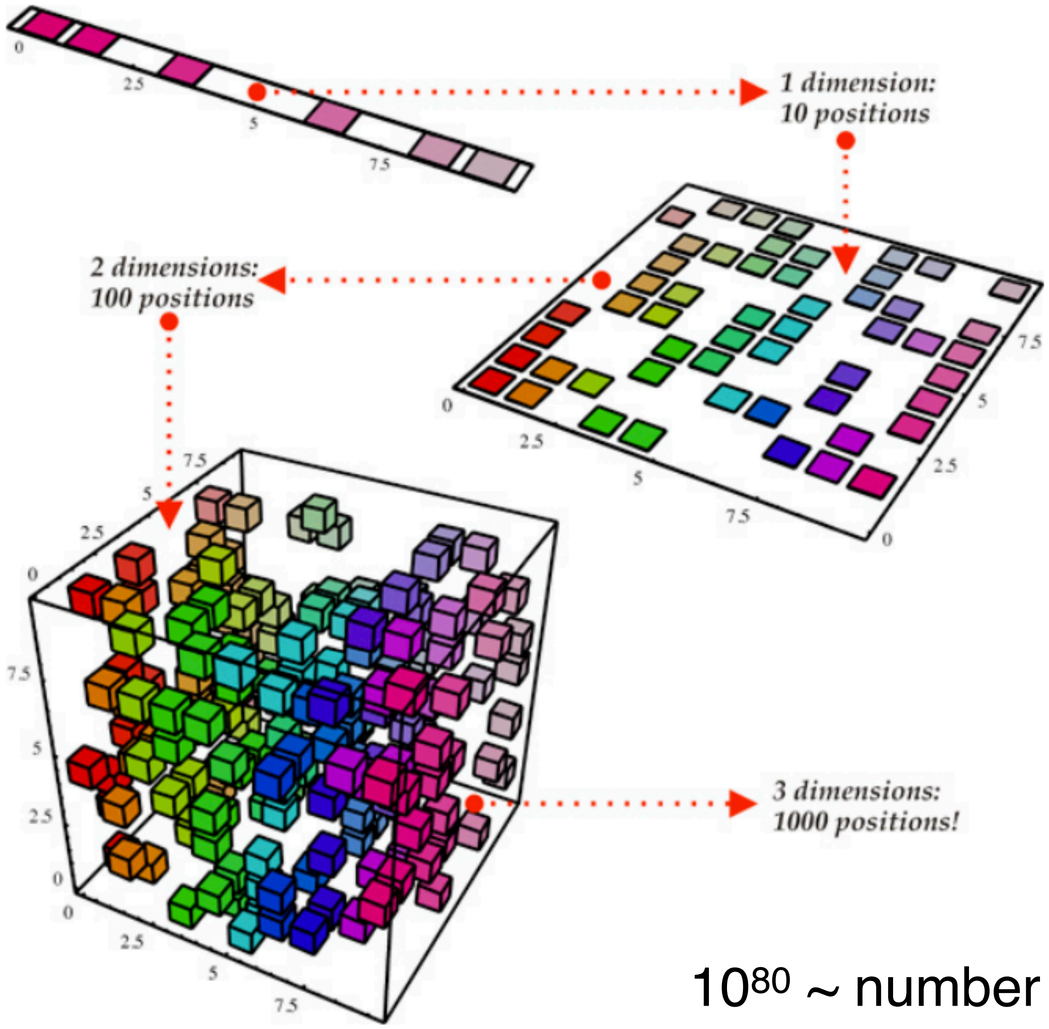
Data preprocessing

The problem: different variables are usually not comparable
(length in m, loudness in dB, cost in hryvnia, …)

Potential remedies:

- Transformation: logarithm, power (e.g.: weight vs linear size)
- Centering (translation): subtract mean, median
- Scaling: divide by range (max-min, standard deviation, …)

Making these choices is a bit of an art. No hard rules. Key: be transparent, reproducible, avoid “p-value hacking”



10^{80} ~ number of particles in the universe

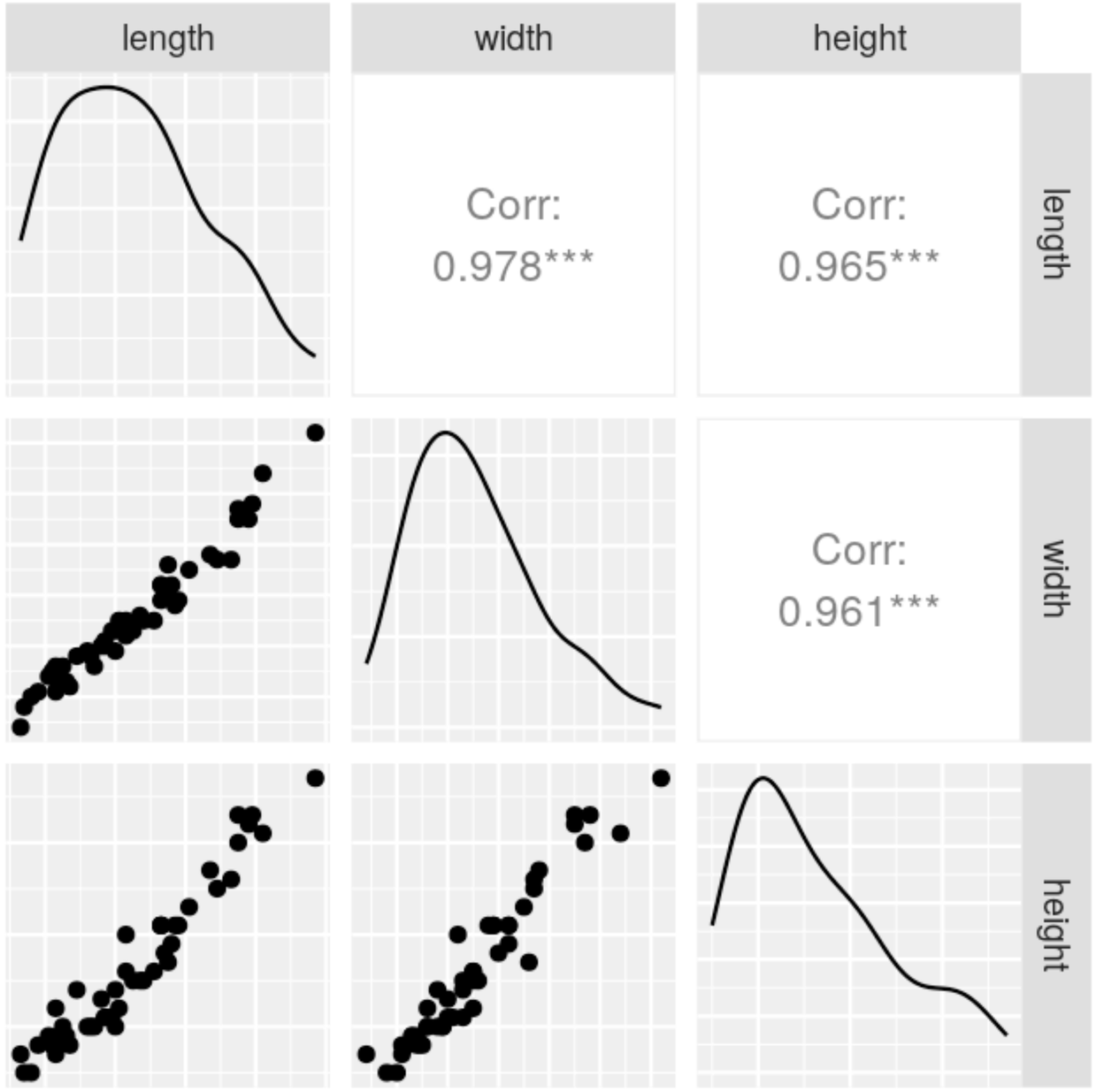
Dimensionality Reduction

Why reduce dimensions?

- Recall the **turtles** data.
  Let's make a pairs plot!

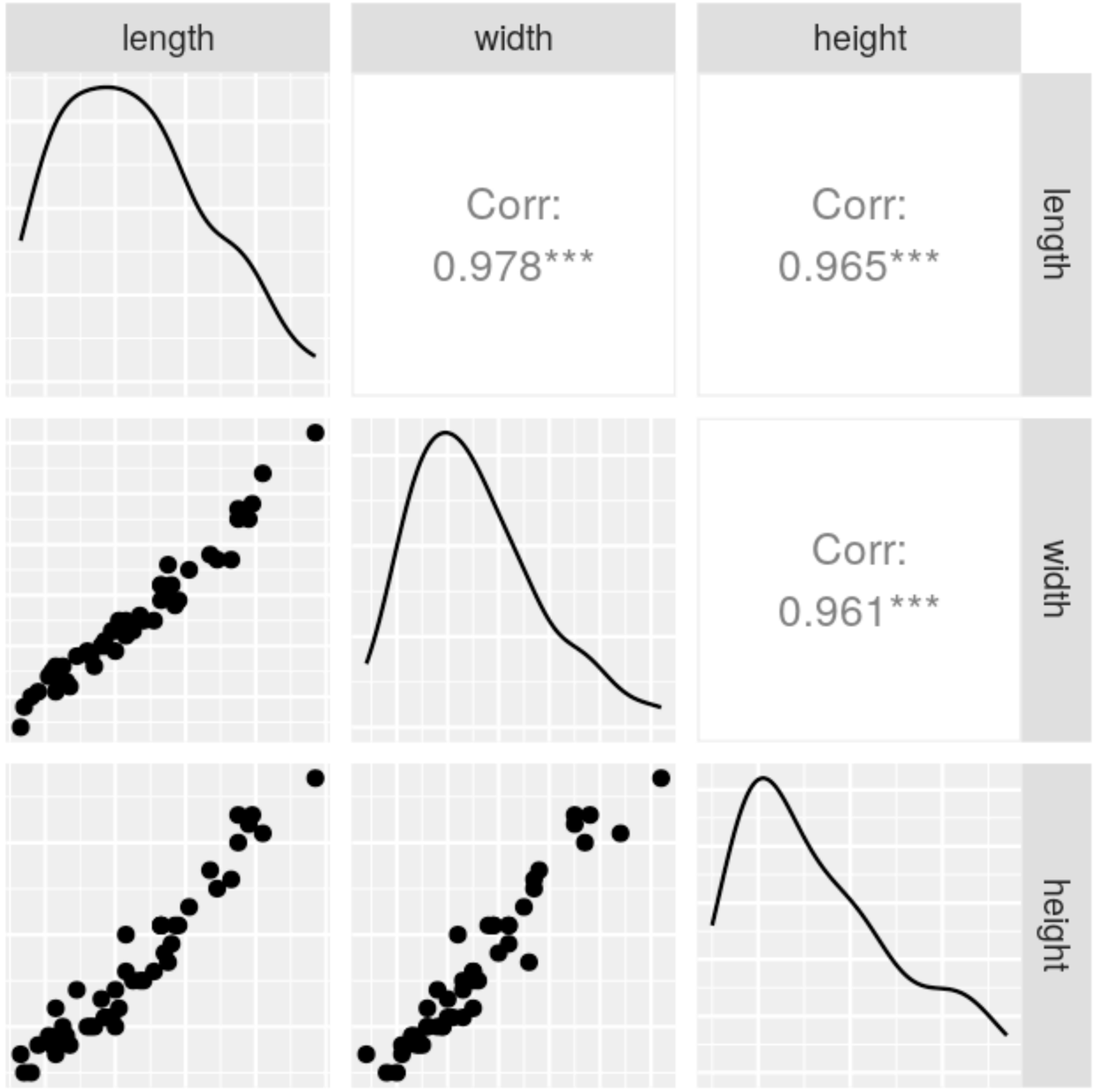
Why reduce dimensions?

- Recall the **turtles** data. Let's make a pairs plot!



Why reduce dimensions?

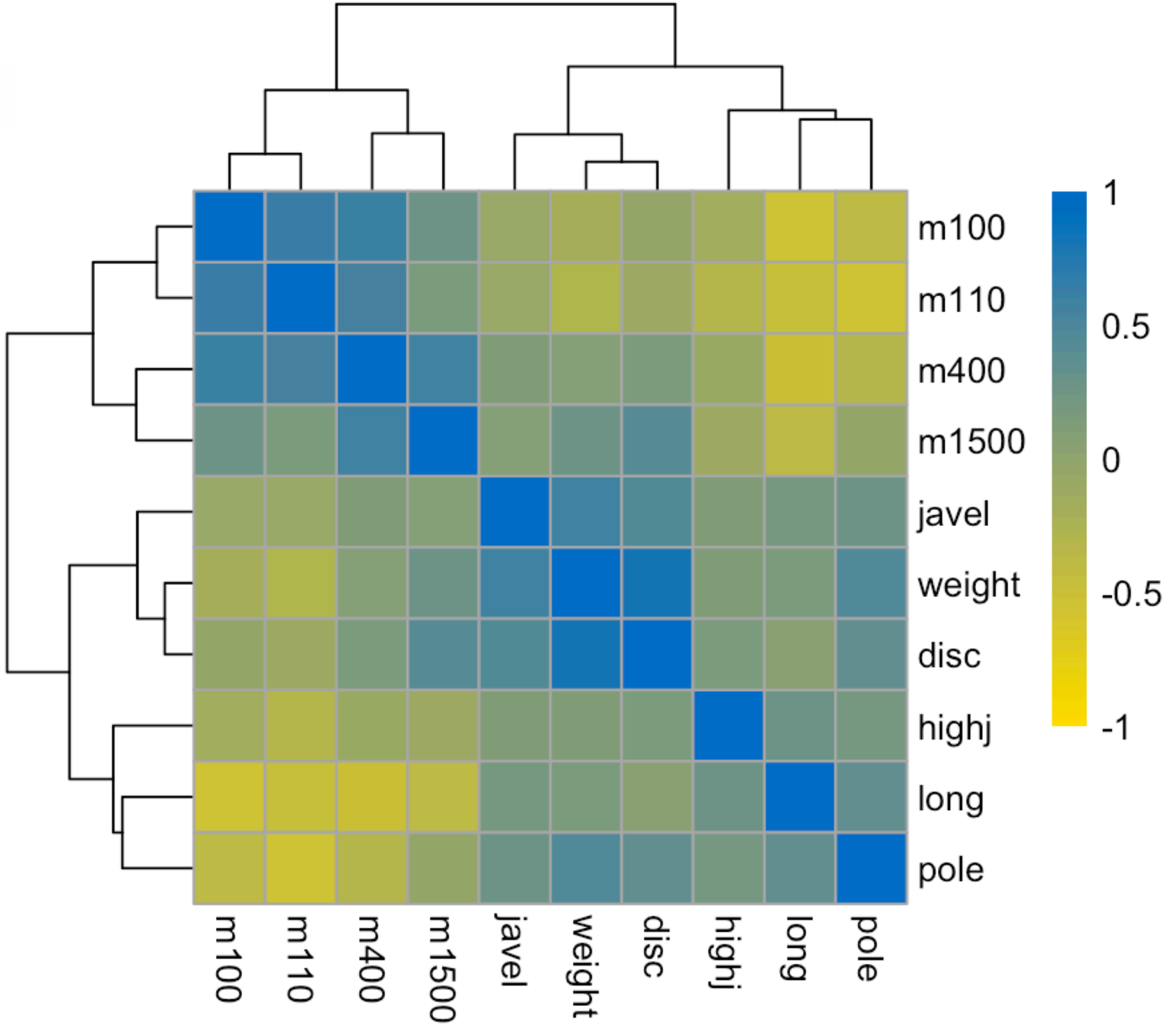
- Recall the **turtles** data. Let's make a pairs plot!



All three of the variables are highly correlated and mostly reflect the same "underlying" variable, which we might interprete as the size of the turtle.

Why reduce dimensions?

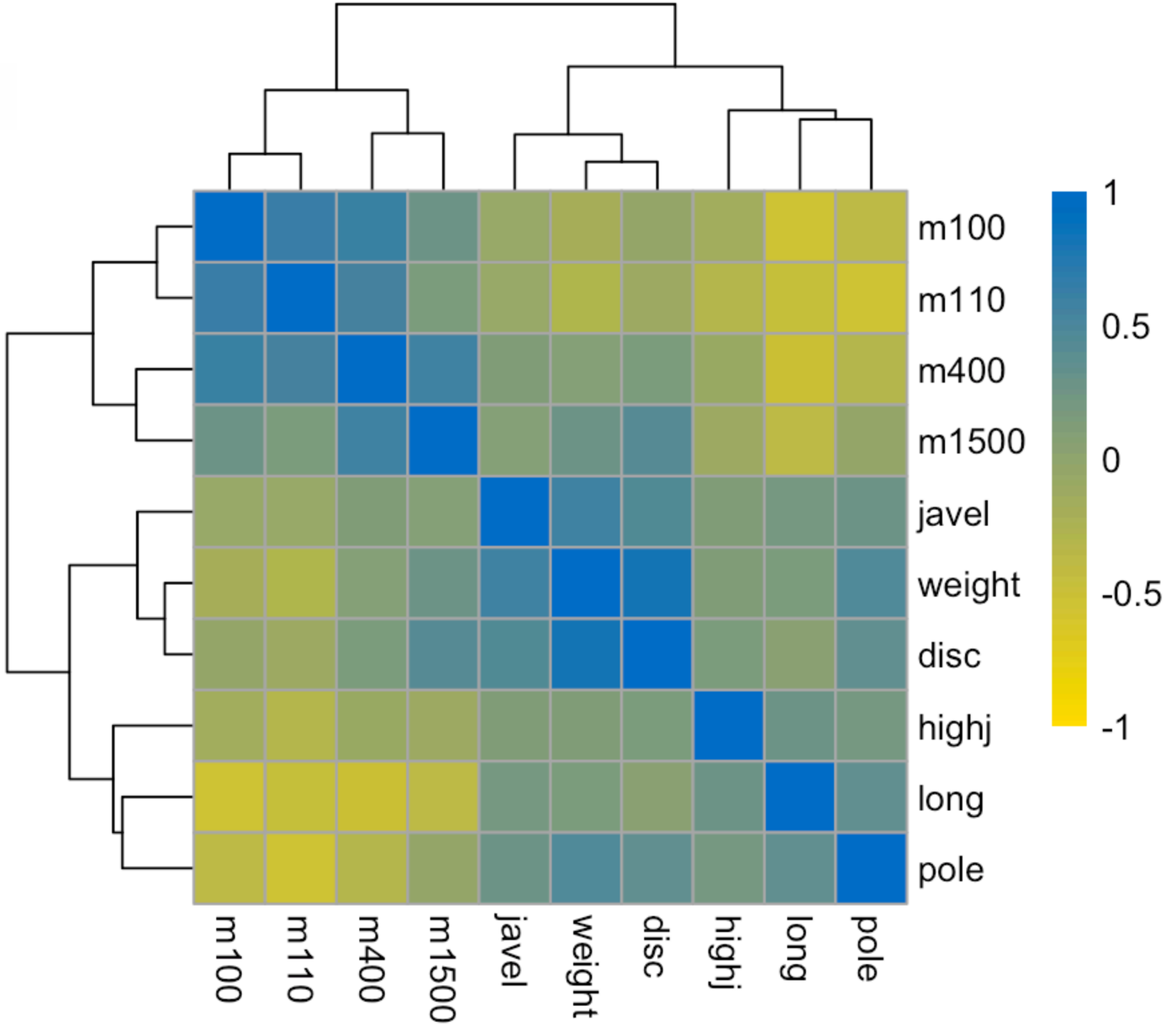
- Recall the athletes (decathlon) data. Look at all pairwise correlations — here we show them as a heatmap.



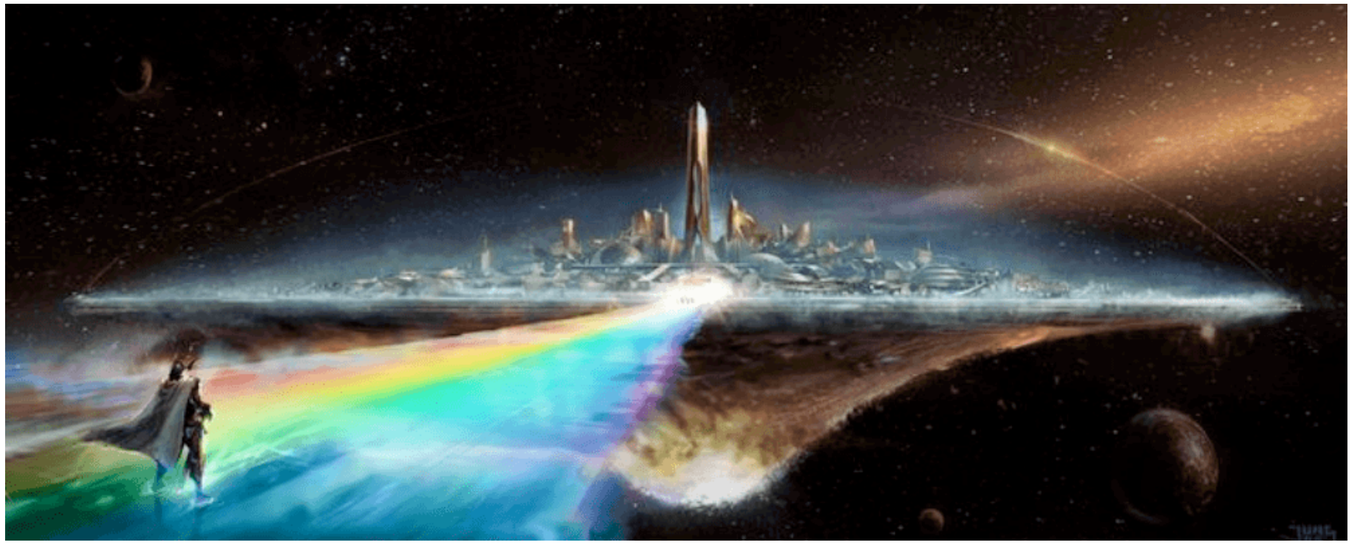
Why reduce dimensions?

- Recall the athletes (decathlon) data. Look at all pairwise correlations — here we show them as a heatmap.

The 10 variables cluster in two or three correlated blocks - 4 for “running”, 4 for “throwing things” and 3 for “jumping”. We might interpret these as underlying “latent factors”



Why reduce dimensions?



- Many biological datasets are high-dimensional. Some also contain substantial measurement noise.
- Low-dimensional manifold hypothesis: the underlying dynamics or variation depends on much fewer, truly important variables, which we do not observe directly, but whose downstream effects we see.
- Utility:
    - Compress the data
    - Increase interpretability
    - Reduce noise
    - Simplify computations, increase (ML) model identifiability

PLOS COMPUTATIONAL BIOLOGY

EDUCATION

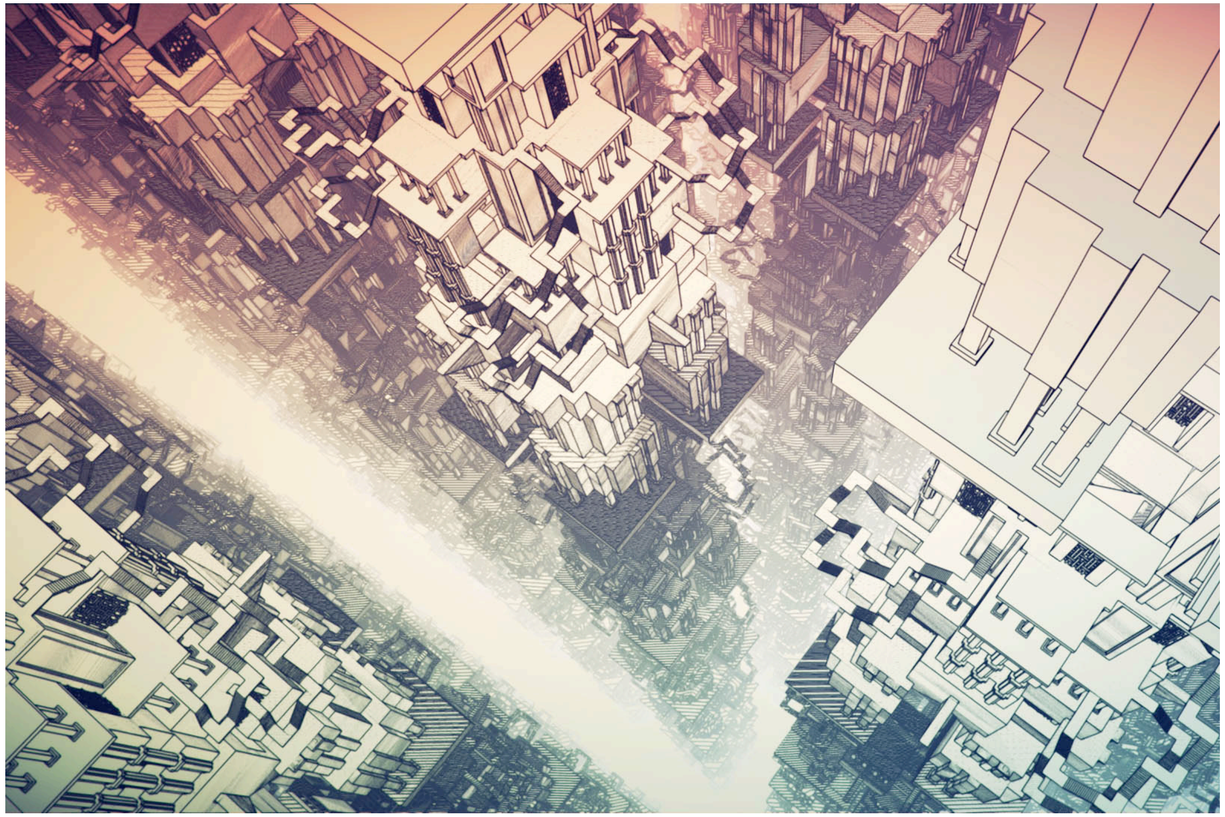
Ten quick tips for effective dimensionality reduction

Lan Huong Nguyen[1], **Susan Holmes**[2]*

1 Institute for Mathematical and Computational Engineering, Stanford University, Stanford, California, United States of America, **2** Department of Statistics, Stanford University, Stanford, California, United States of America

* susan@stat.stanford.edu

How to reduce dimensions?



- Find “reasonable” (linear, smooth, …) mathematical functions that map (data) points from the original high-dimensional space into lower dimensions (e.g., 2D screen)
- This may be called “unsupervised learning”: infer latent (hidden) variables from “unlabeled” data.
- Prototypical version: Principal Component Analysis (PCA)

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 the1930s. Statisticians used it to summarize a battery of psychological tests run on the same subjects, by constructing overall scores that summarize many variables at once.
- This idea of principal scores inspired the name Principal Component Analysis.

Karl Pearson

Harold Hotelling

Projecting 2D-data on a line

- In general, dimension reduction implies that we **lose information.**
- Our goal is to **keep as much information as possible**.
- We can use this to choose among the many ways to project a point cloud on a line (1D), plane (2D), etc.

Projecting 2D-data on a line

- In general, dimension reduction implies that we **lose information.**
- Our goal is to **keep as much information as possible**.
- We can use this to choose among the many ways to project a point cloud on a line (1D), plane (2D), etc.

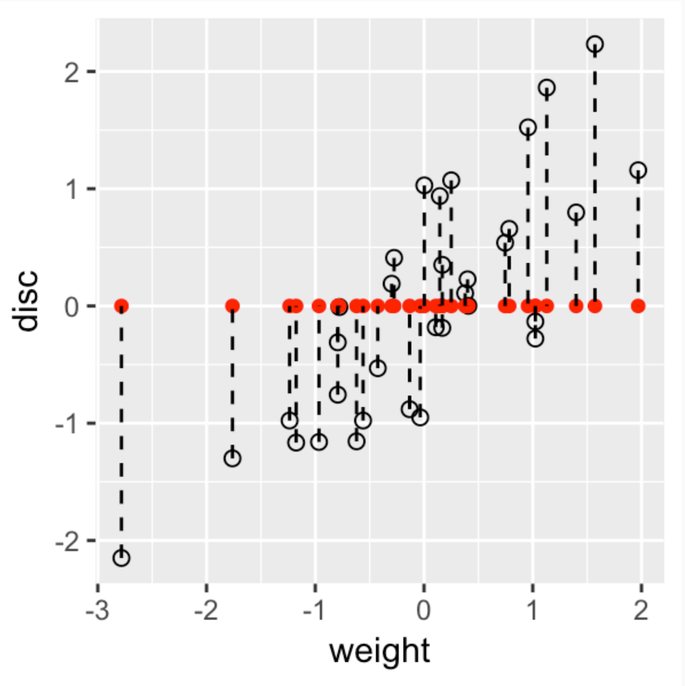


Figure 7.6: Scatterplot of two variables showing the projection on the horizontal x axis (defined by $y = 0$) in red and the lines of projection appear as dashed.

Projecting 2D-data on a line

- In general, dimension reduction implies that we **lose information.**
- Our goal is to **keep as much information as possible**.
- We can use this to choose among the many ways to project a point cloud on a line (1D), plane (2D), etc.

If we just project on some of the original coordinate axes, for instance the *x*-axis, we lose all of its information about the other(s).

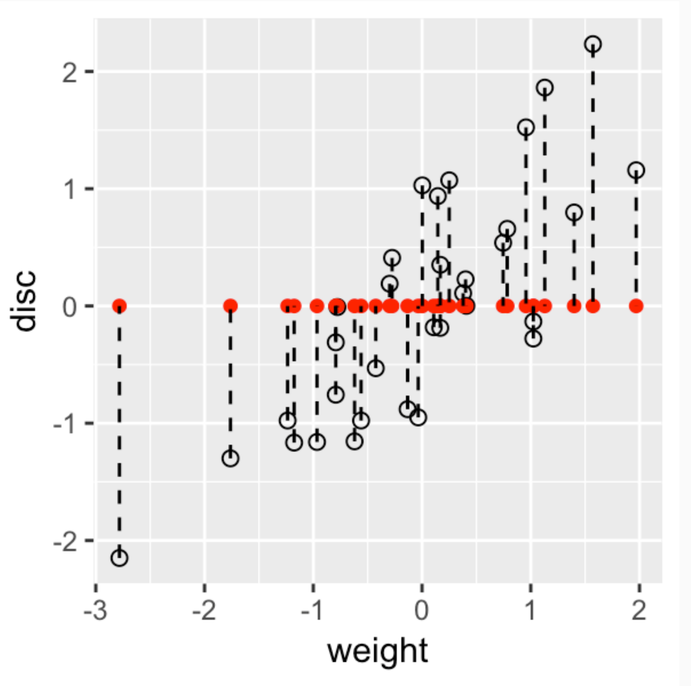


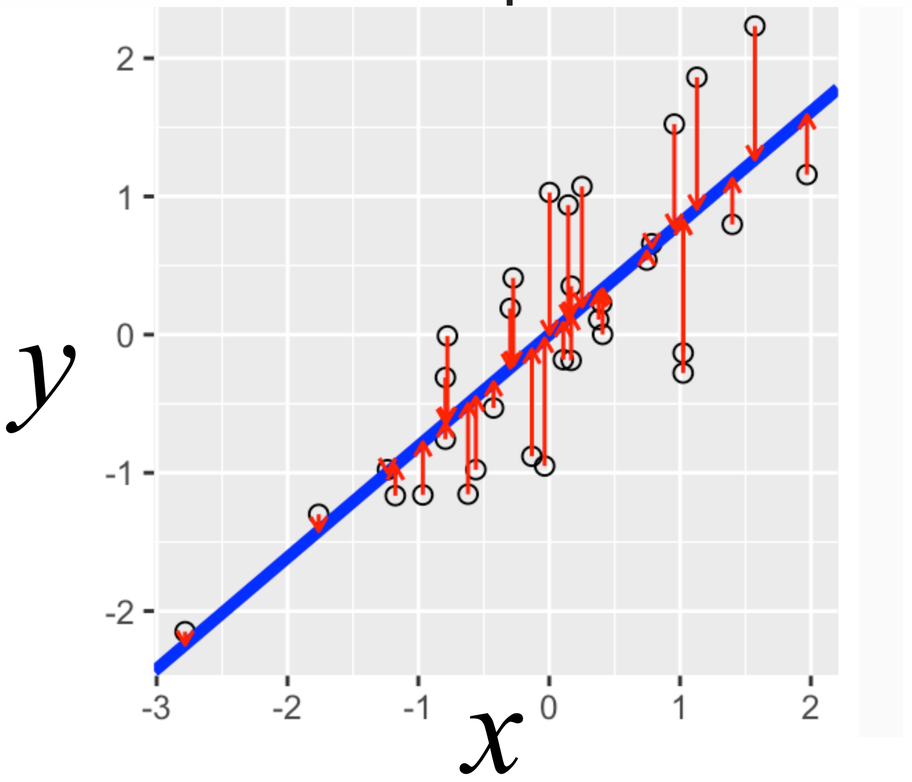
Figure 7.6: Scatterplot of two variables showing the projection on the horizontal x axis (defined by $y = 0$) in red and the lines of projection appear as dashed.

Regression Line

- One of the most widely known method of projecting 2D data on a line is **regression.**
- Regressing y variable on x variable minimizes the vertical distances (red bars)
- **Linear regression** is a **supervised** method that gives preference minimizing the residual sum of squares in one direction (direction of the response variable).

$$y_i = a + bx_i + \varepsilon_i$$

$$\sum_i \varepsilon_i^2 \rightarrow \textbf{min}$$

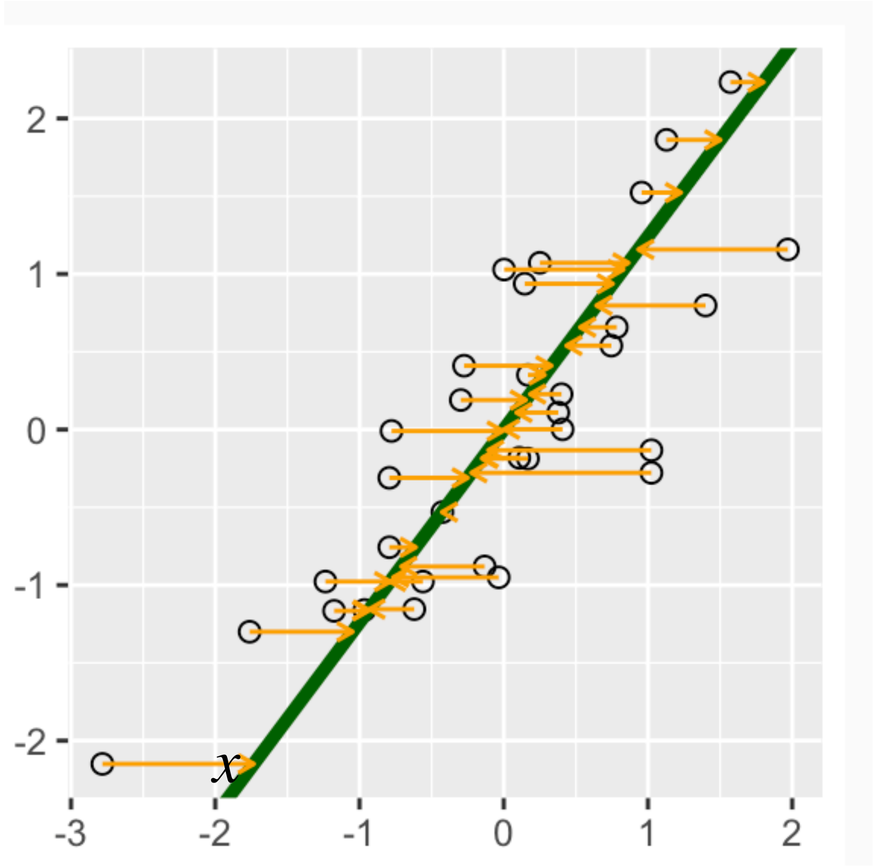


Regression Line

Of course we can also regress in another direction, i.e., regress x on y:

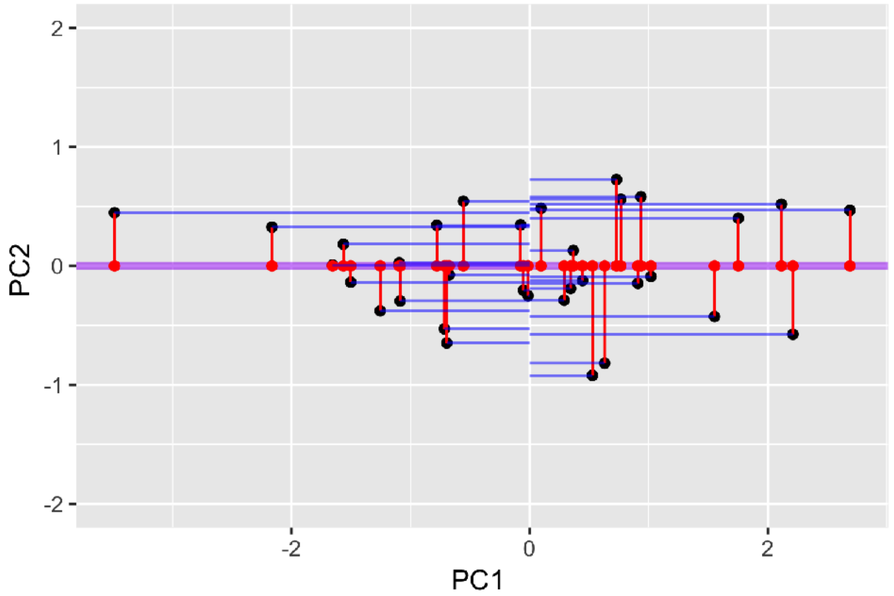
$$x_i = a + by_i + \varepsilon_i$$

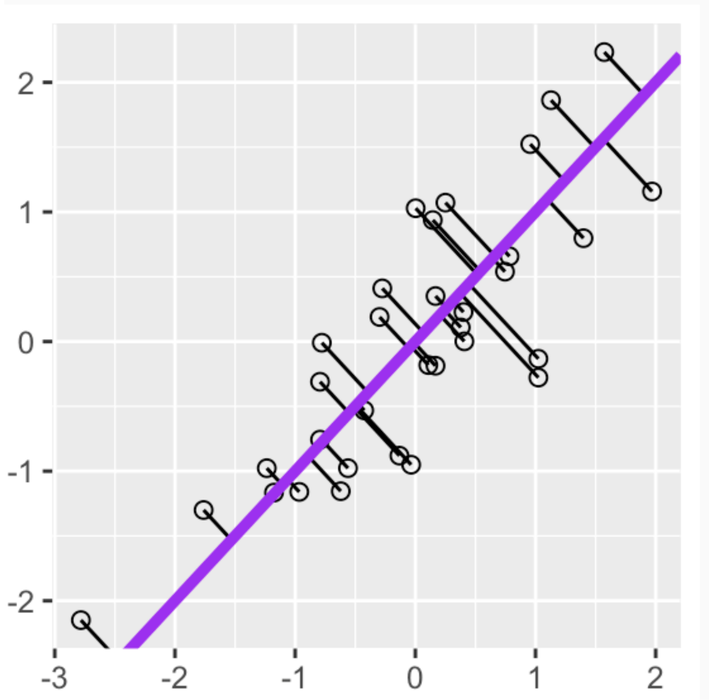
$$\sum_i \varepsilon_i^2 \rightarrow \textbf{min}$$



Principal Component Line

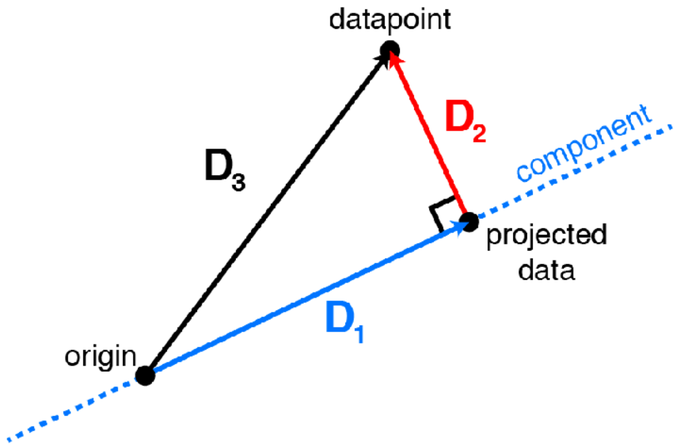
- Line **minimizing error in both horizontal and vertical directions**
- We are in fact minimizing the diagonal projections onto the line from each point





Pythagoras' theorem

- **Total variance:** sum of squares of the distance of the points to the center of the point cloud
- This is also called the **inertia of the point cloud**. It can be decomposed into **the sum of the squares of the projections onto the line plus the variances along that line**.
- For a given point cloud, **minimizing the projection distances is equivalent maximizing the variance** along that line. We can use either criterion to define the first principal component.



$$D_3^2 = D_1^2 + D_2^2$$

initial variance = remaining variance + lost variance

$$\|a_i\|^2 = \|w_i c\|^2 + \|a_i - w_i c\|^2$$

this is constant — maximize this — **or** — minimize this

Linear Combinations

- The PC line we found in the previous section could be written $PC = \frac{1}{2}x + \frac{1}{2}y$
- **Principal components are linear combinations of variables that were originally measured,** they provide a new coordinate system.
- This is analogous to what you do when. making a healthy juice mix, you can follow a recipe.

$$V = 2 \times \text{Beets} + 1 \times \text{Carrots} + \frac{1}{2}\,\text{Gala} + \frac{1}{2}\,\text{GrannySmith} + 0.02 \times \text{Ginger} + 0.25\,\text{Lemon}$$

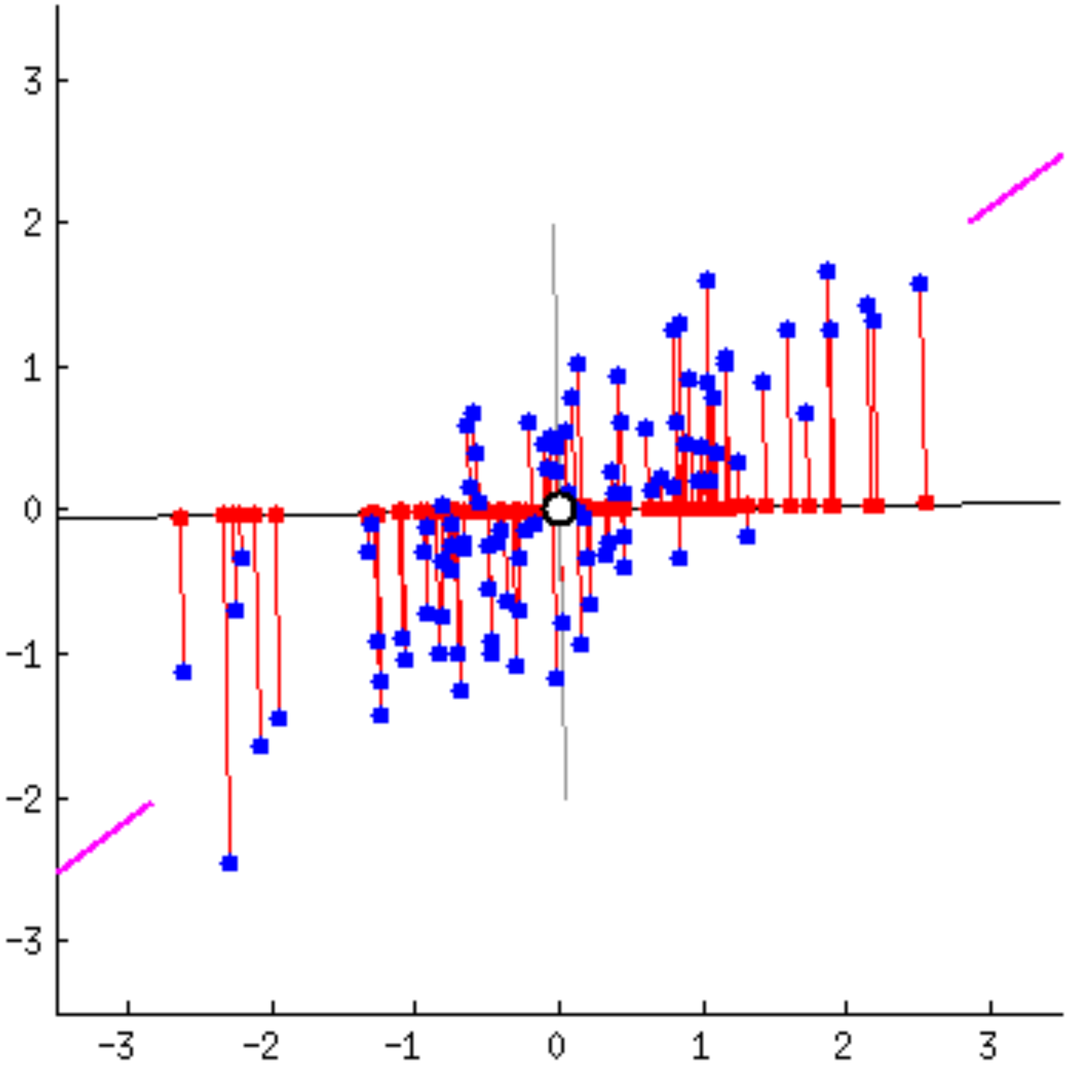
ingredients

- 2 pounds beets (about 6 medium), trimmed, peeled, cut into 1" pieces
- 1 pound carrots (about 4 large), trimmed, peeled, cut into 1" pieces
- 1 Gala or Empire apple (about 8 ounces), cored, cut into 1" pieces
- 1 Granny smith apple (about 8 ounces), cored, cut into 1" pieces
- 1 3" piece fresh ginger, peeled, chopped into 1" pieces
- 3 tablespoons fresh lemon juice

The above recipe is a linear combination of individual ingredients.

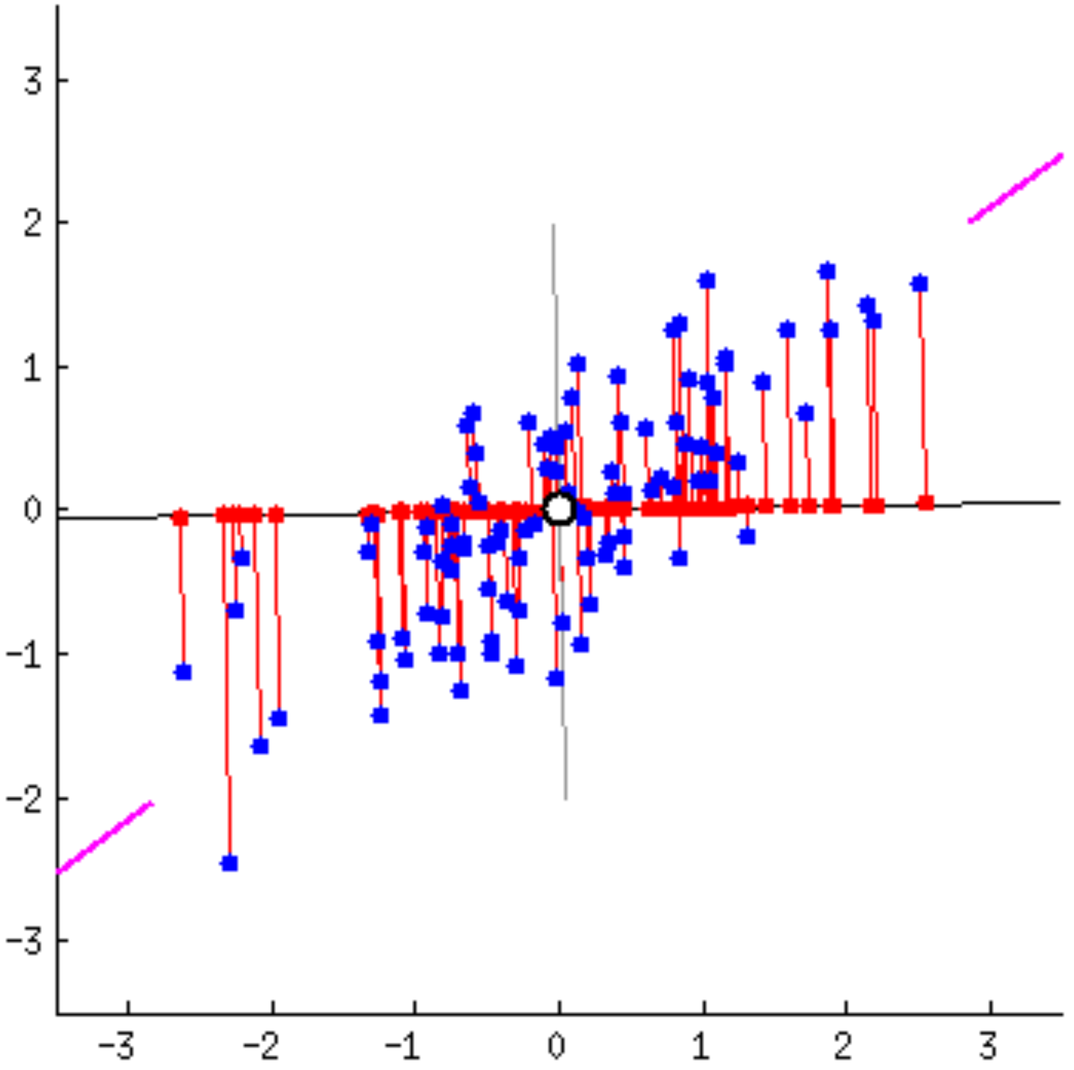
The juice mix PC would be a new variable, with coefficients (2,1, 0.5,0.5,0.02,0.25) for each original ingredient, called loadings.

Optimal Line



Source
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Optimal Line



Source
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Optimal Line

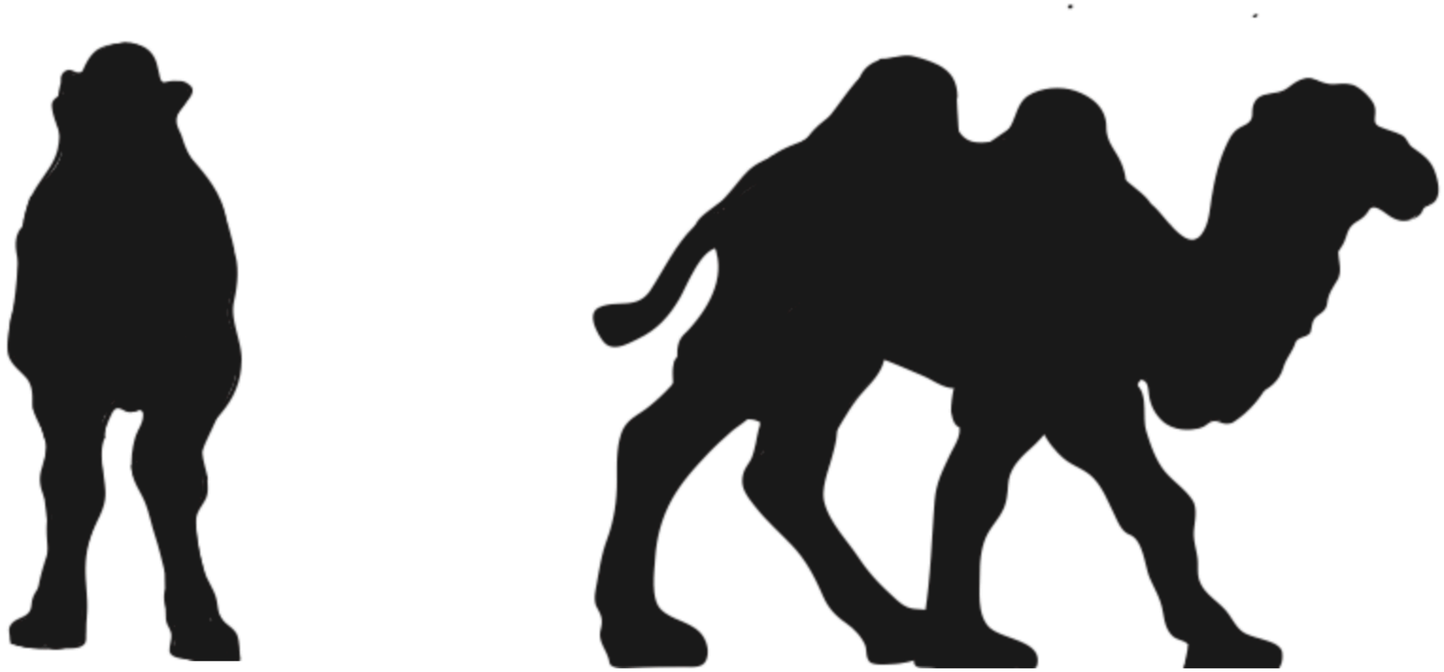
- In higher dimensions (> 2), **a linear combination of variables still defines a line.**
- As before, there are many ways to choose lines onto which we project the data.
- In PCA, we use the fact that the total sums of squares of the distances between the points and the origin can be **decomposed into the distance to the line and the variance along the line**.
- We saw that the principal component **minimizes the distance to the line**, and it also **maximizes the variance of the projections** along the line.
- **Why is this a good idea?** Let's look at another example of a projection from 3D to 2D, demonstrating what happens in human vision

Good Projections

- What is this?

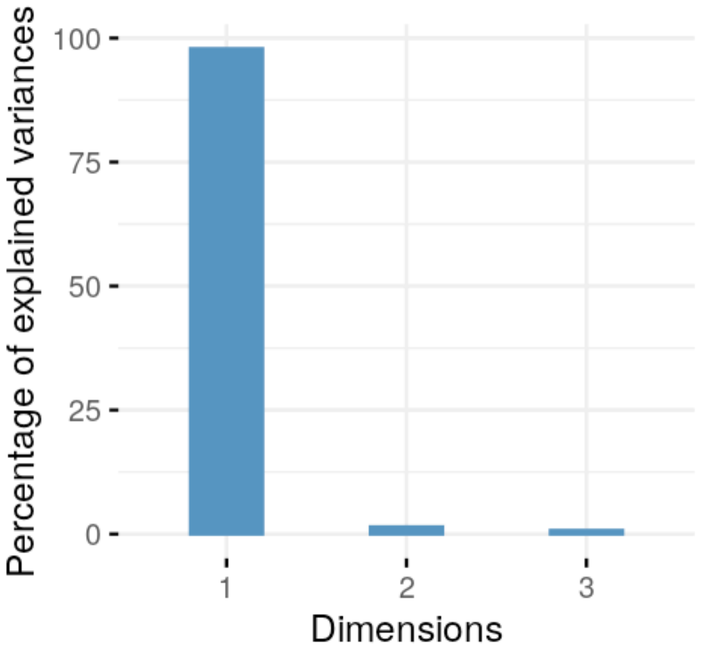
Mystery Image

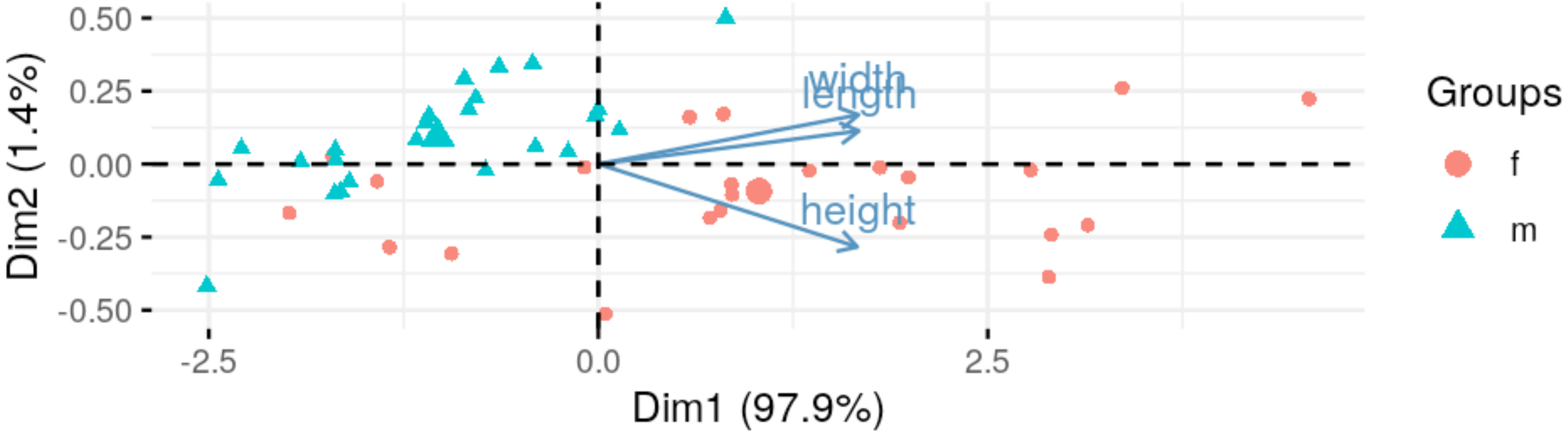
Good Projections



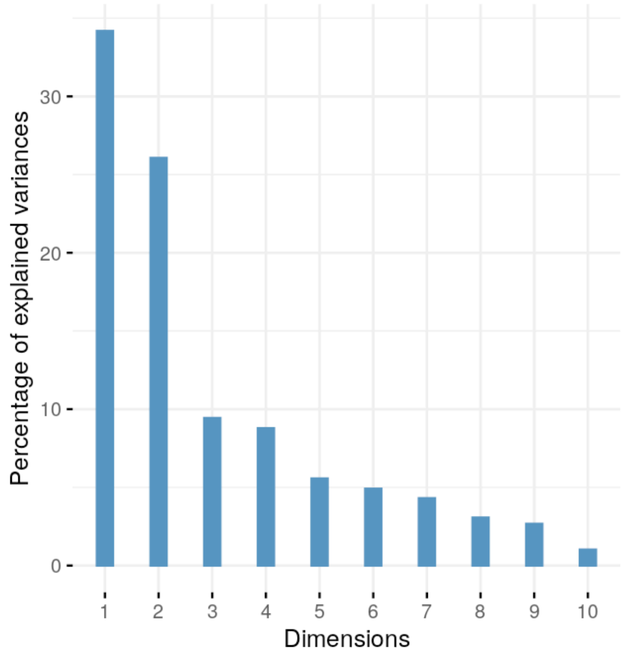
- Which projection do you think is better?
- It's the **projection that maximizes the area of the shadow.** An equivalent criterion is the sums of squares of the distances between points in the projection: we want to see as much of the variation as possible. **That's what PCA does.**

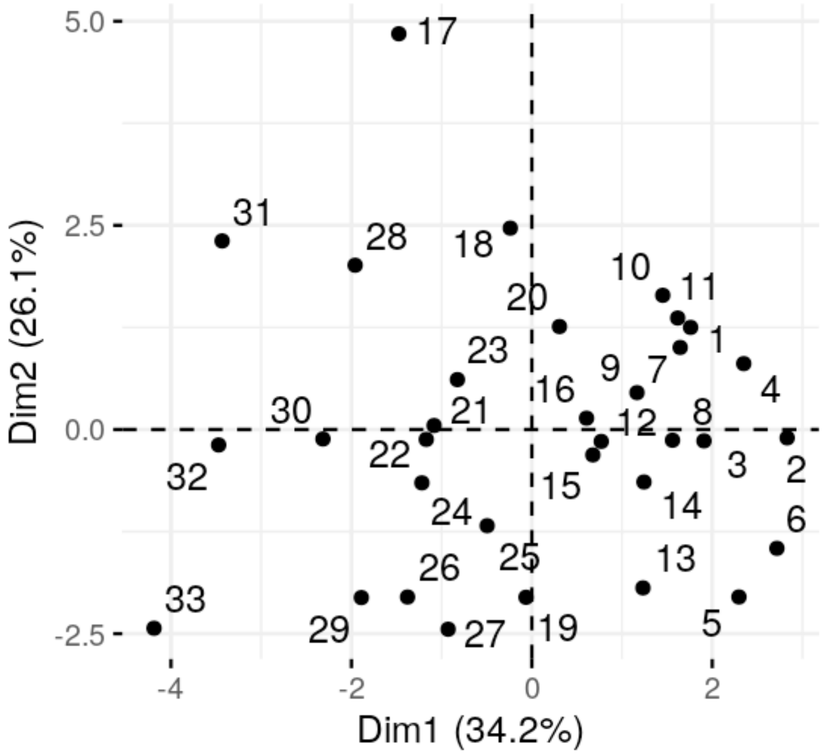
PCA of the turtles data

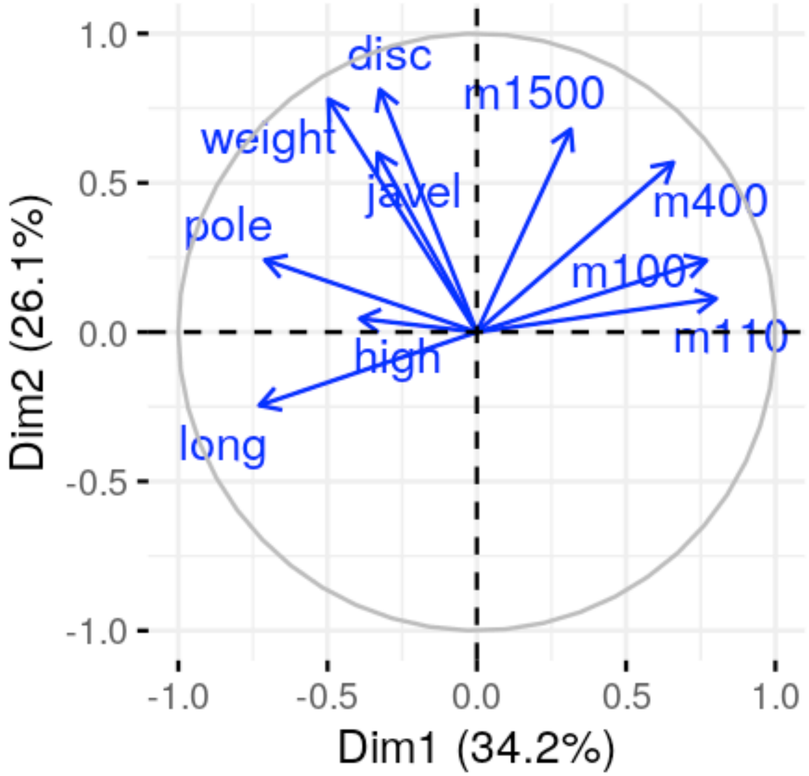




PCA of the athletes data







How does it work?

- **First PC accounts for as much of the variance as possible.** Then iterate: each **successive PC explains the highest remaining variance possible under** the constraint that it's orthogonal to the preceding ones.
- Solution can computed rapidly and exactly using basic linear algebra: Singular Value Decomposition (SVD), Matrix Diagonalization (eigenvectors).
- See MSMB book (or many other textbooks).

Running PCA in R

Wine Dataset

- This dataset contains **chemical measurements on different wines.**
- We also have a information 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 highly recommend looking up this stats-stackexchange post which is also about wines!

https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

	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("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)
apply(wine_scaled, 2, mean)
```

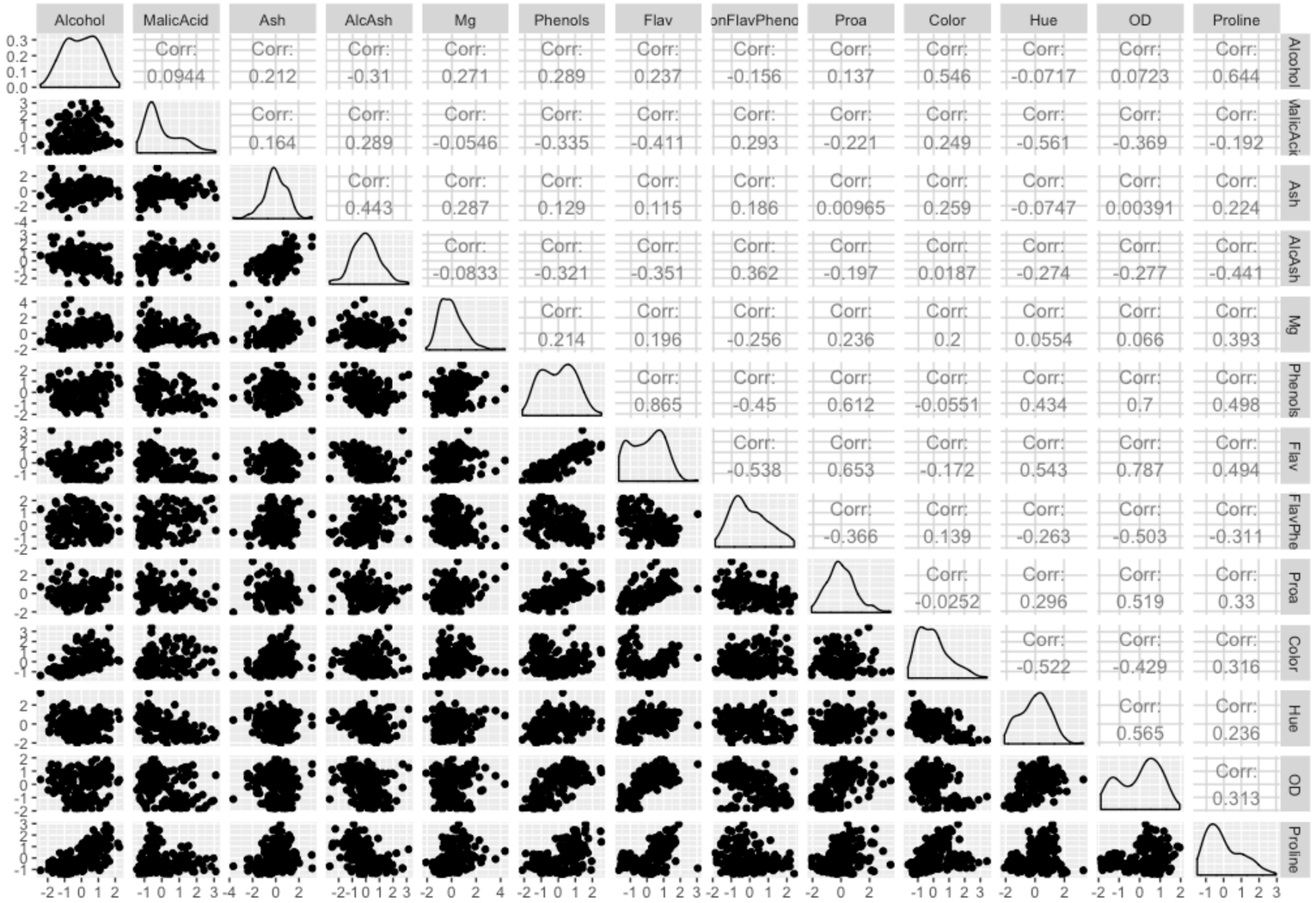
```
##        Alcohol      MalicAcid            Ash        AlcAsh            Mg
##  -8.591766e-16  -6.776446e-17   8.045176e-16  -7.720494e-17  -4.073935e-17
##        Phenols           Flav NonFlavPhenols          Proa          Color
##  -1.395560e-17   6.958263e-17  -1.042186e-16  -1.221369e-16   3.649376e-17
##            Hue             OD        Proline
##   2.093741e-16   3.003459e-16  -1.034429e-16
```

```
apply(wine_scaled, 2, sd)
```

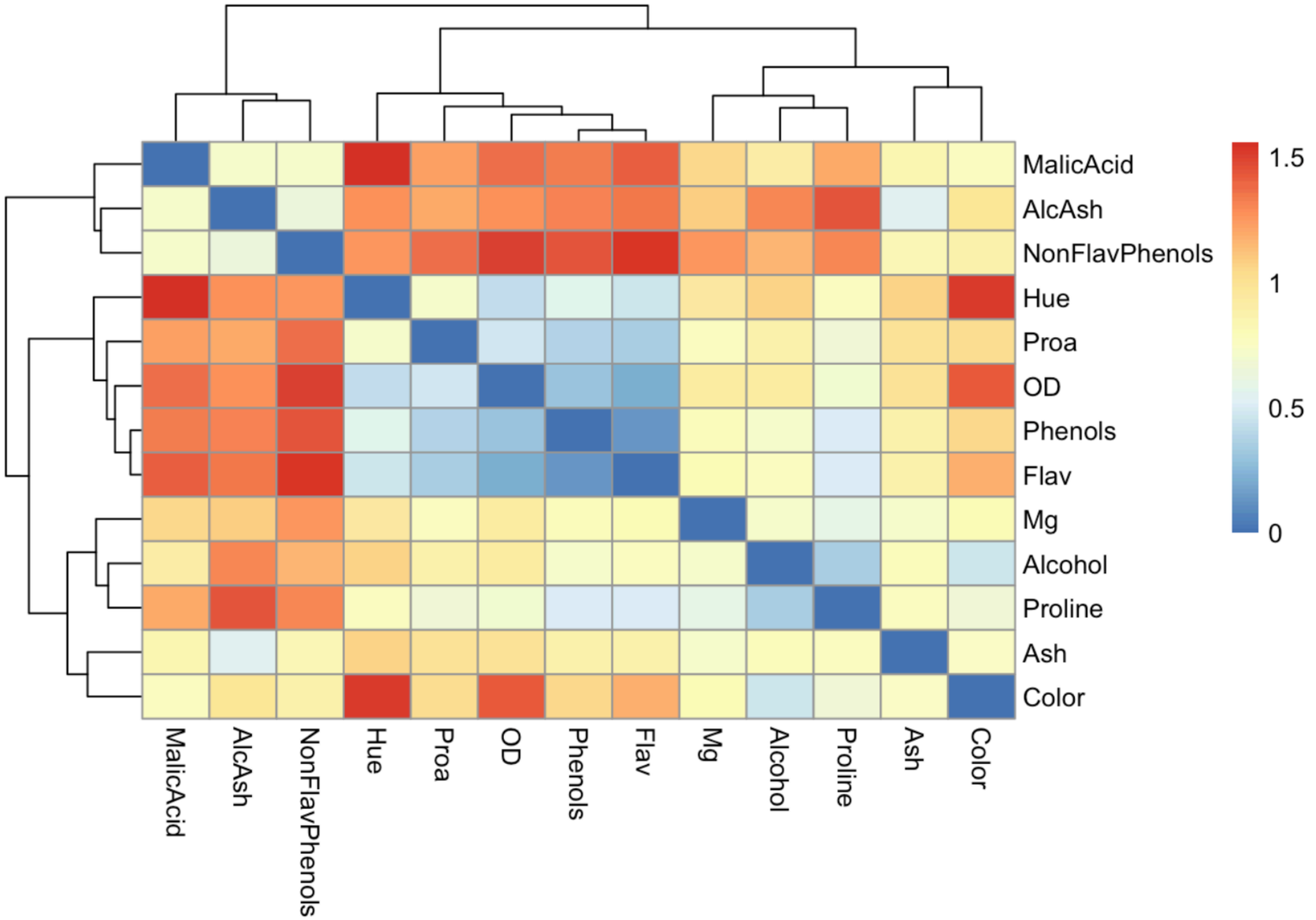
```
##        Alcohol      MalicAcid            Ash        AlcAsh            Mg
##              1              1              1              1              1
##        Phenols           Flav NonFlavPhenols          Proa          Color
##              1              1              1              1              1
##            Hue             OD        Proline
##              1              1              1
```

2D summaries

```
wine_scaled <- as.data.frame(wine_scaled)
GGally::ggpairs(wine_scaled)
```



```r
library(pheatmap)
pheatmap(1 - cor(wine))
```



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 ahead of time!

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

```
class(winePCA)
```

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

```
names(winePCA)
```

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

- Note that we set nf = 5, that is 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

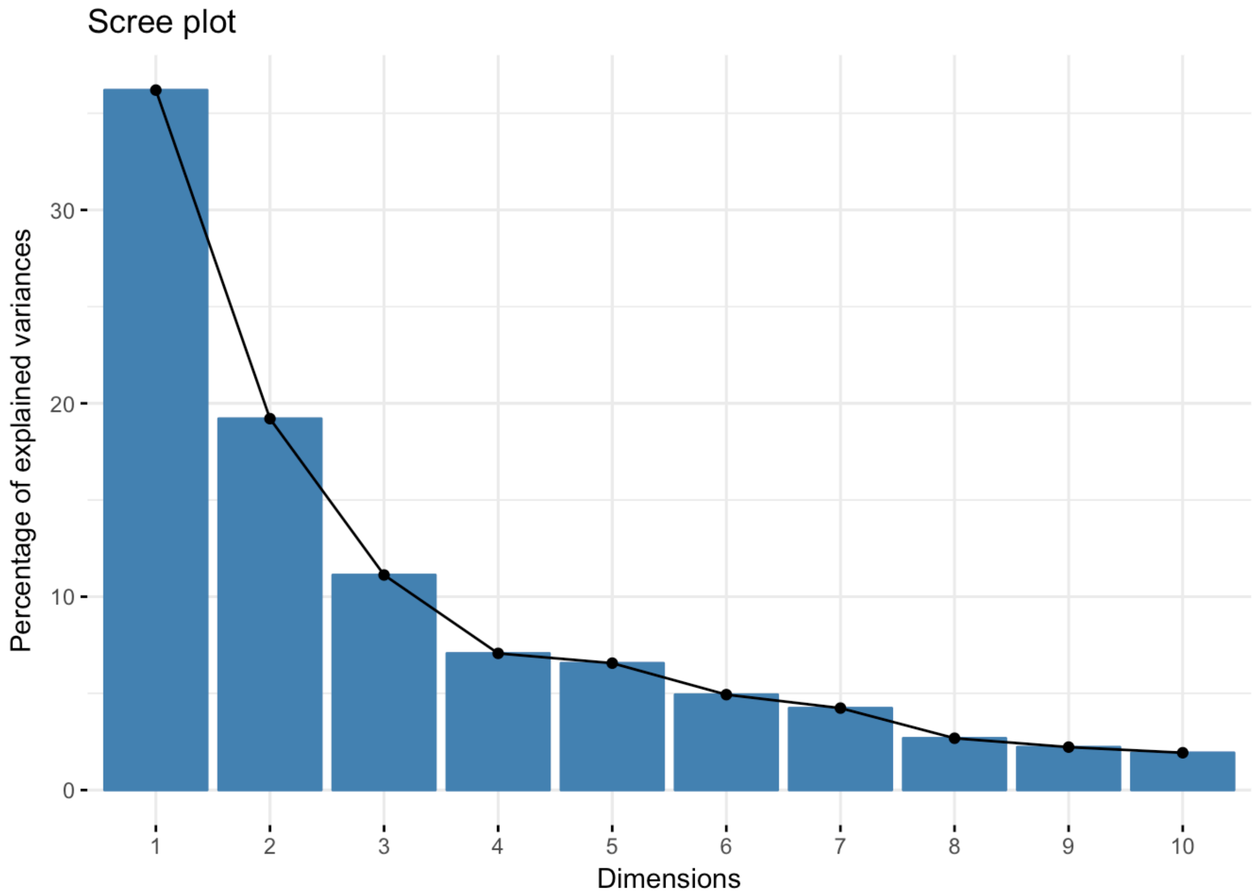
```
names(winePCA)
```

```
##  [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)

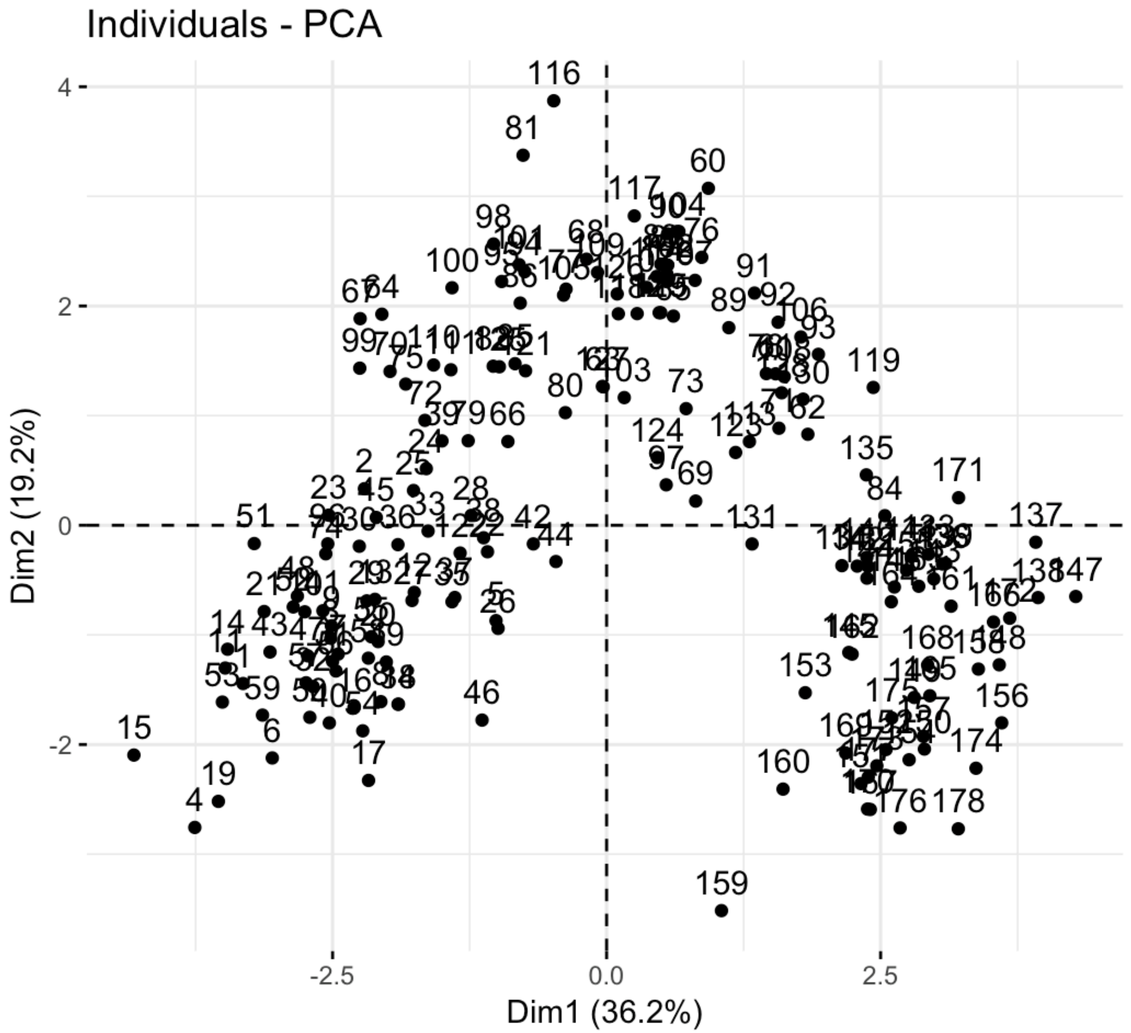
Scree Plot

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



Sample Projection

```
fviz_pca_ind(winePCA) +
  coord_fixed()
```

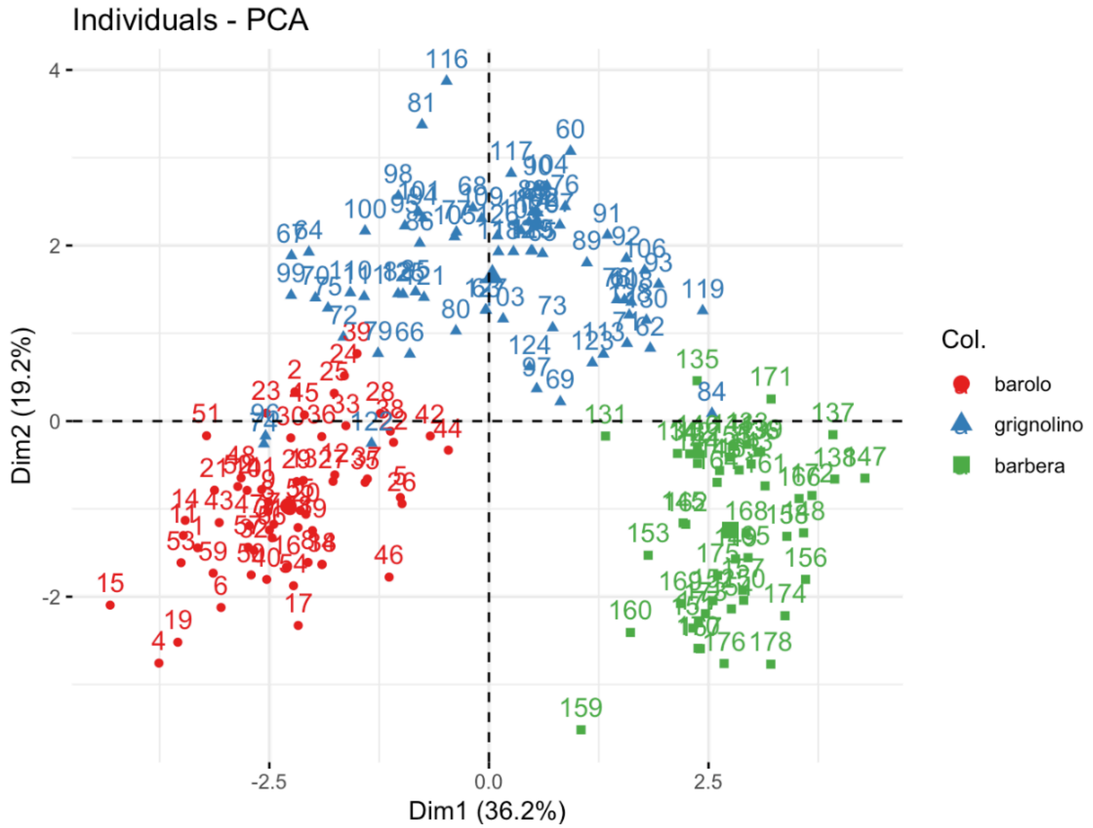


Sample Projection with Coloring by Covariates

```
load("wineClass.RData")
table(wine.class)
```

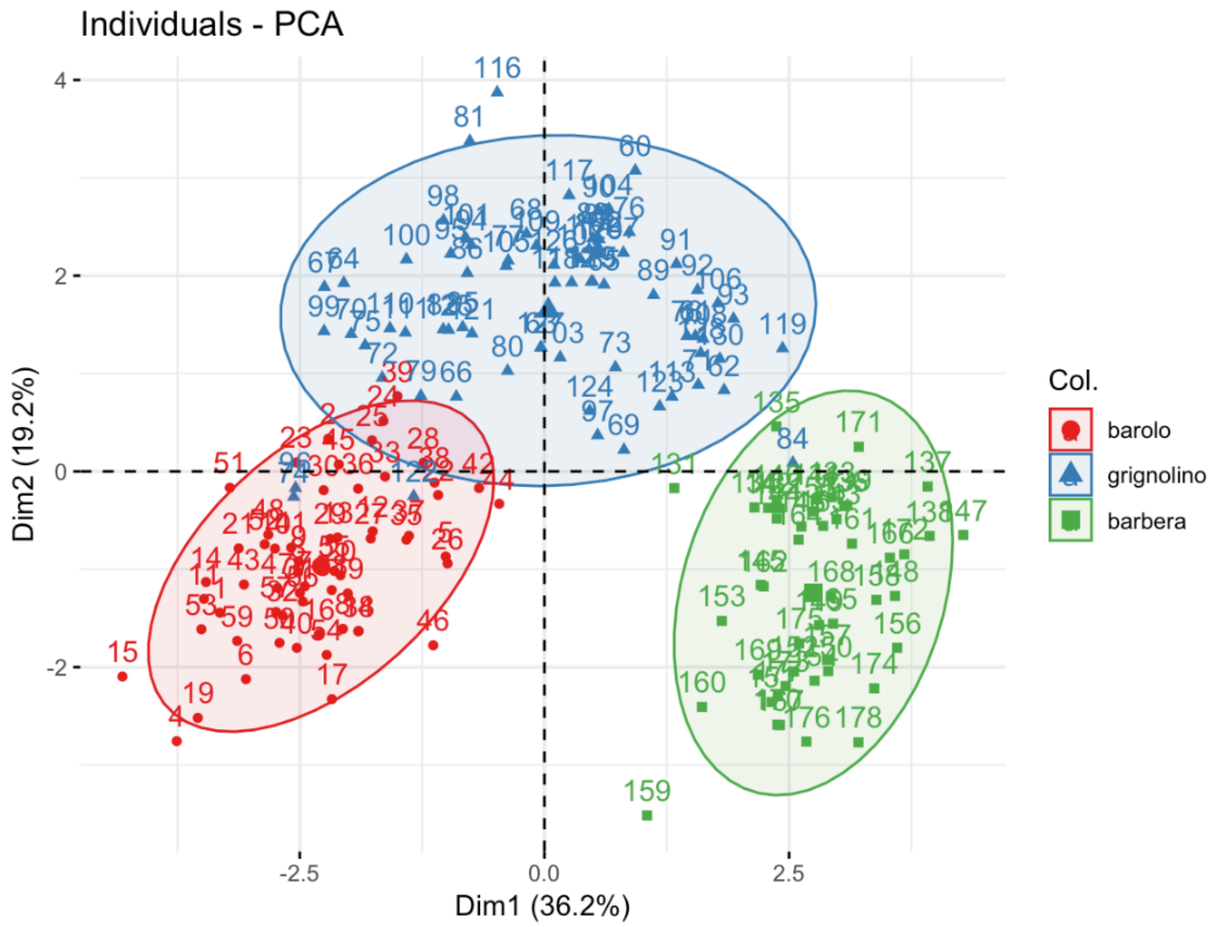
```
## wine.class
##     barolo grignolino    barbera
##         59         71         48
```

```
fviz_pca_ind(winePCA, col.ind = wine.class,
  palette = c("#E41A1C", "#377EB8", "#4DAF4A")) +
  coord_fixed()
```



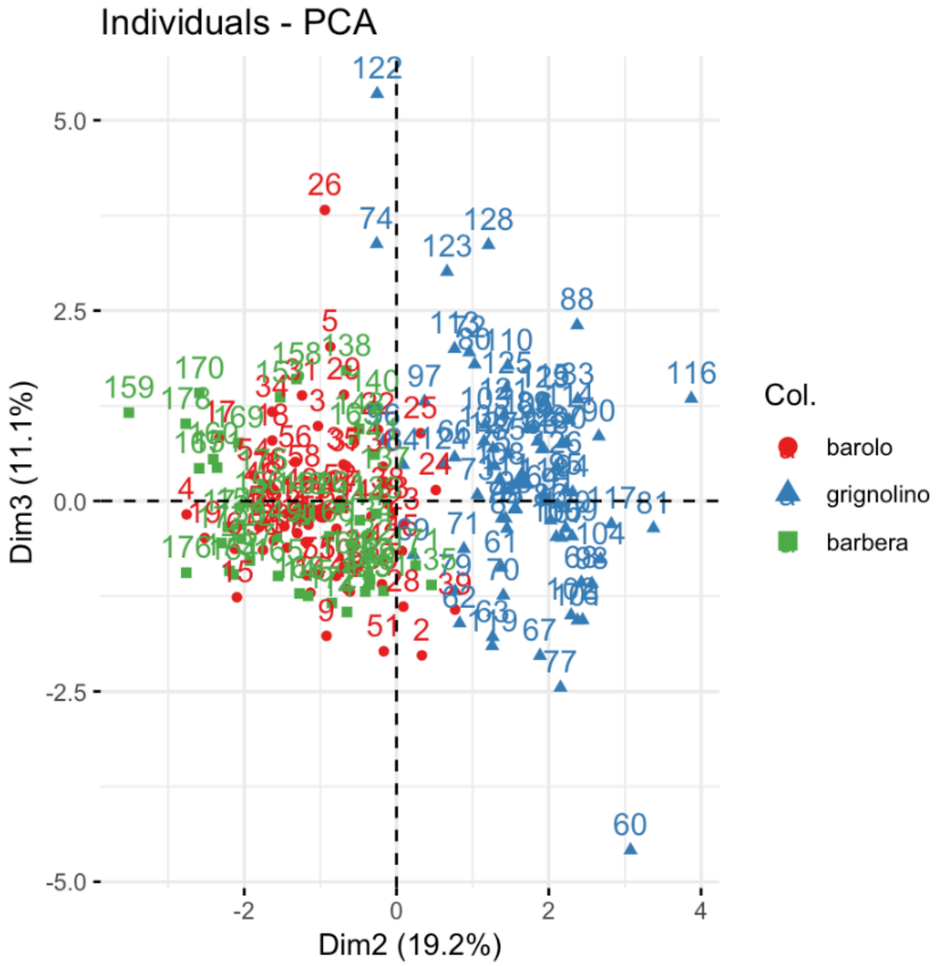
Including Ellipses

```
fviz_pca_ind(winePCA, col.ind = wine.class,
  palette = c("#E41A1C", "#377EB8", "#4DAF4A"),
  addEllipses = TRUE, ellipse.level = 0.9) +
  coord_fixed()
```



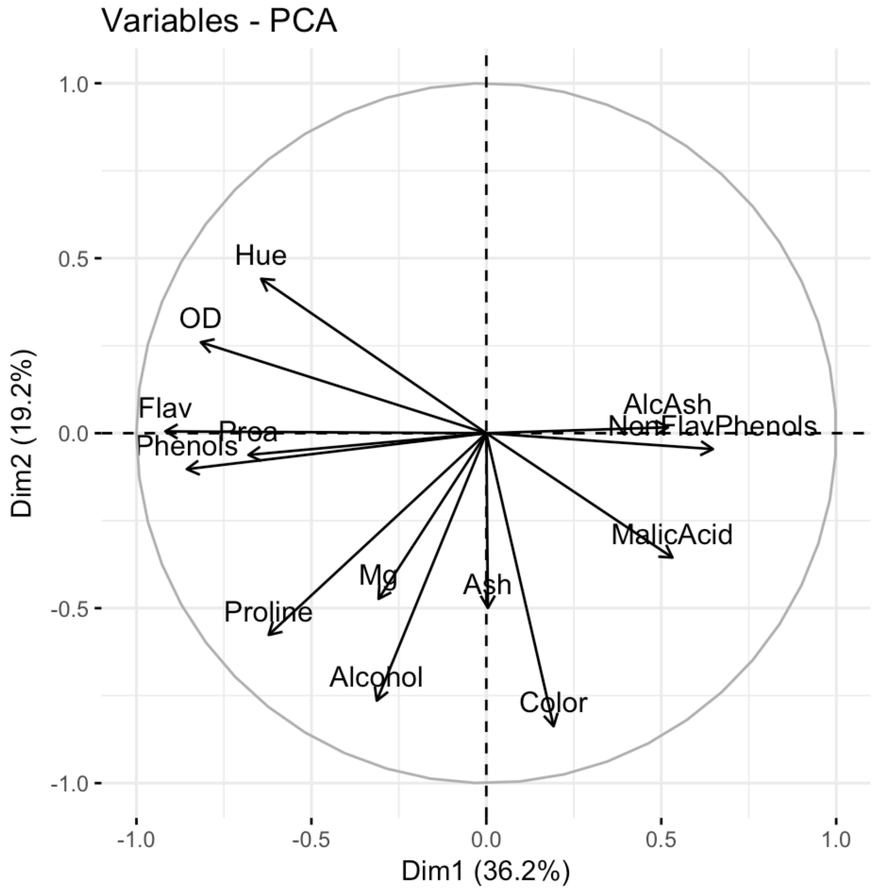
Sample Projection on Other Axes

```
fviz_pca_ind(winePCA, axes = 2:3, col.ind = wine.class,
  palette = c("#E41A1C", "#377EB8", "#4DAF4A")) +
  coord_fixed()
```



Correlation Circle for Variables

```
fviz_pca_var(winePCA)
```

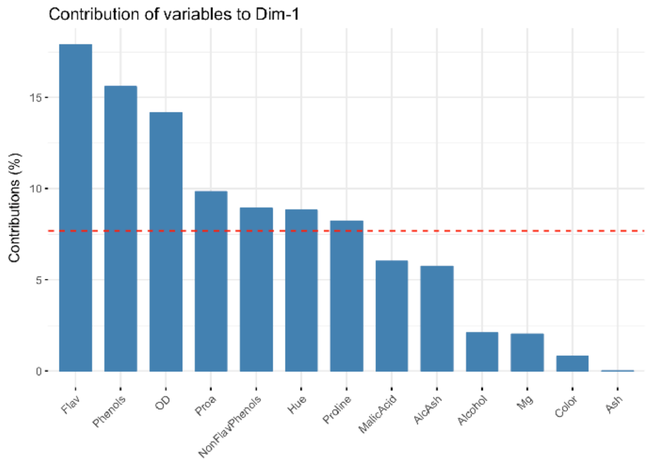


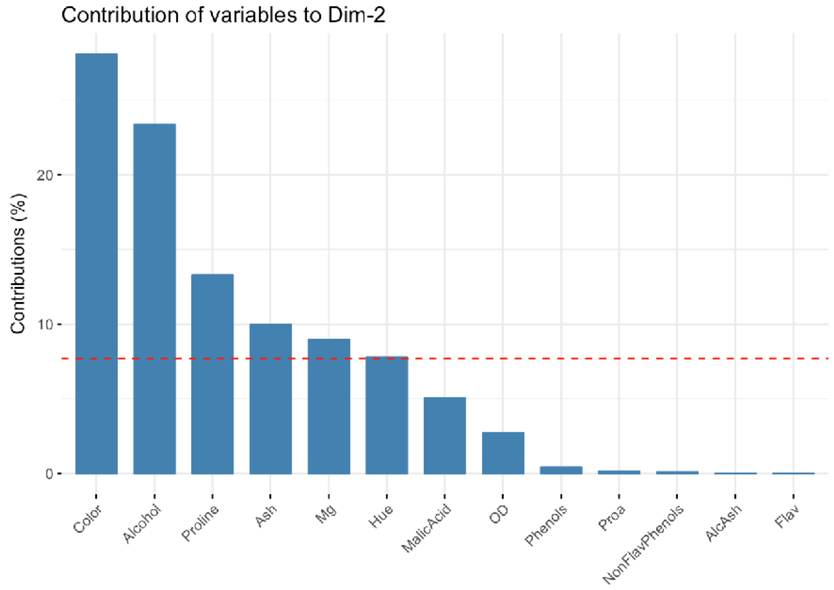
Variable Contribution to PCs

- 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)
```

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

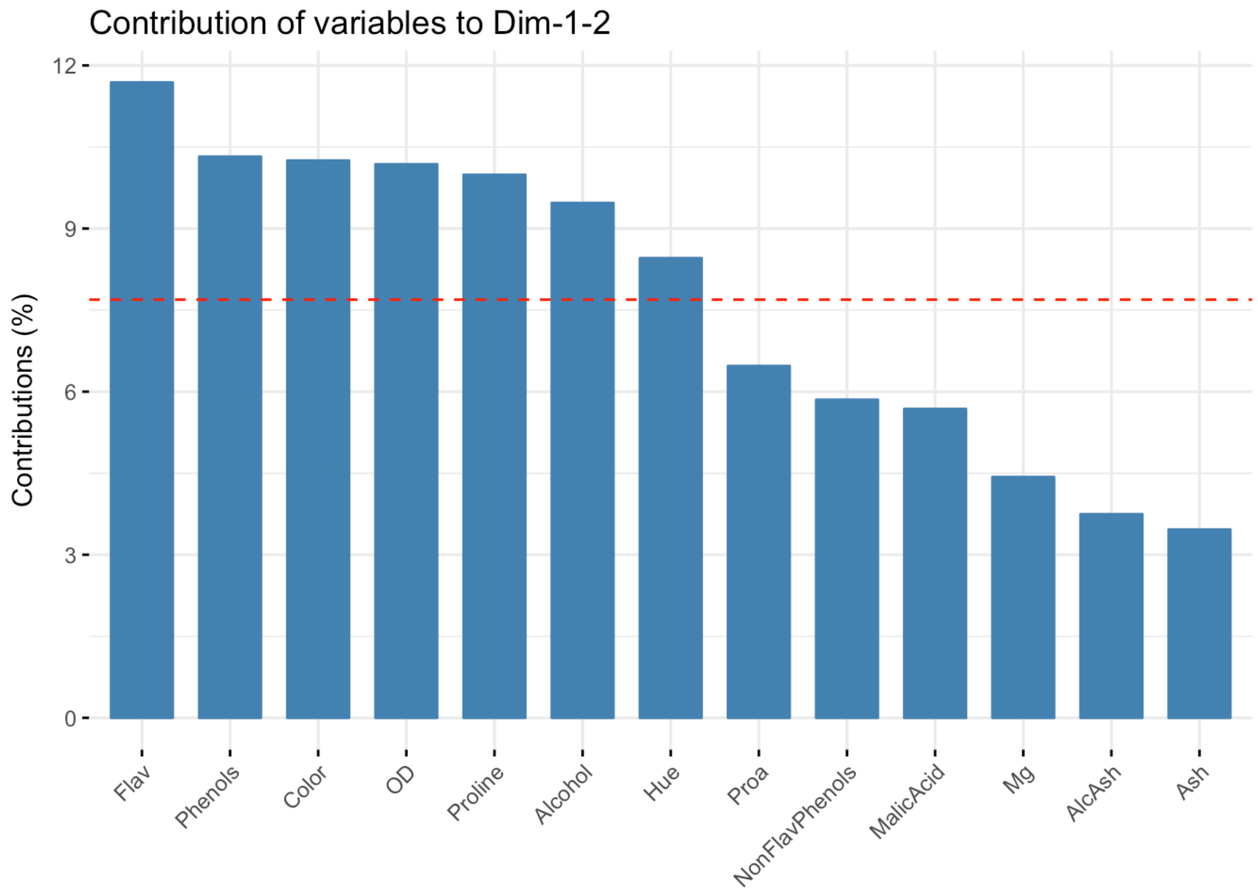




The red dashed line on the graph above indicates the expected average contribution, as if the contribution of the variables was even.

Variable Contribution to PCs

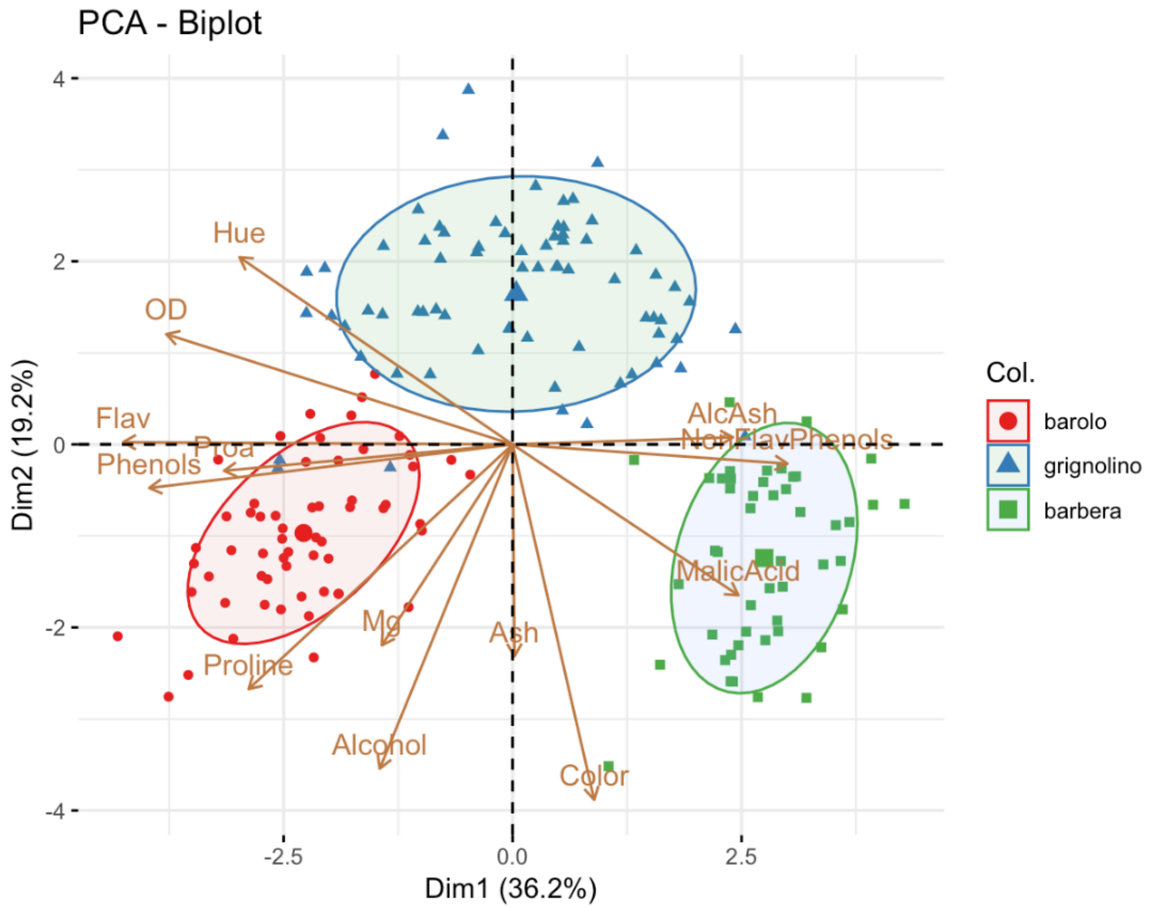
```
fviz_contrib(winePCA, choice = "var", axes = 1:2)
```



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

PCA Biplot with Everything Together

```
fviz_pca_biplot(
  winePCA, geom = "point",
  col.ind = wine.class,
  col.var = "#c07d44",
  addEllipses = TRUE, ellipse.level = 0.7) +
  coord_fixed() +
  scale_color_brewer(palette = "Set1")
```



Further Reading

- The best way to deepen your understanding of SVD is to read Chapter 7 of *Strang (2009).*
- Complete textbook on PCA and related method, *Mardia, Kent, and Bibby (1979)*, is a standard text that covers all multivariate methods in a classical way, with linear algebra and matrices.
- *Jolliffe (2002)* is a booklong treatment of everything to do with PCA with extensive examples.
- Multi-dimensional scaling, t-SNE, UMAP; autoencoders.
- Chapter 14 of the The Elements (Hastie, Tibshirani, Friedmann)