Introduction

R for Data Science

Juan R Gonzalez
juanr.gonzalez@isglobal.org

BRGE - Bioinformatics Research Group in Epidemiology
ISGlobal - Barcelona Institute for Global Health
http://brge.isglobal.org

Credits

R for Data Science: http://r4ds.had.co.nz/index.html

Outline of the course

- Day 1: Introduction (1h)
  - Bioinformatics Research Group in Epidemiology (https://brge.isglobal.org)
  - Course Material
  - PCA for big data
  - Master Thesis proposal(s)
- Day 1: Reproducible Research (1h)
  - knitr
  - R markdown

- Day 2: Tidyverse - Data wrangling (1h)
  - Data wrangling
  - Tibbles
  - Data import
  - Relational data
  - Strings, Factors (not covered)
- Day 2: Tidyverse - Data transformation (1h)
  - Filter rows
  - Arrange rows
  - Select columns
  - Add columns
  - Grouped summaries
  - Grouped mutates
- Day 3: Tidyverse - Data visualization (1h)
  - Exploratory data analysis
  - Advanced graphics
  - Learning more
- Day 3: Visualization Large Datasets (1h)
  - Genomics: GViz
  - bigvis

Course Material and tasks

- Material available at https://github.com/isglobal-brge/course_tidyverse
- Tasks
  - Daily exercises (Slides + Moodle)
  - Final exercise (Moodle)

Principal component analysis (PCA) in data science

- PCA is one of the most commonly used method to visualize large data sets.
- It aims to reduce the dimensionallity into a couple of principal axes that capture most of the observed data variability.
- PVA is related to the Single Valude Decomposition (SVD) problem.

Example: Genomics

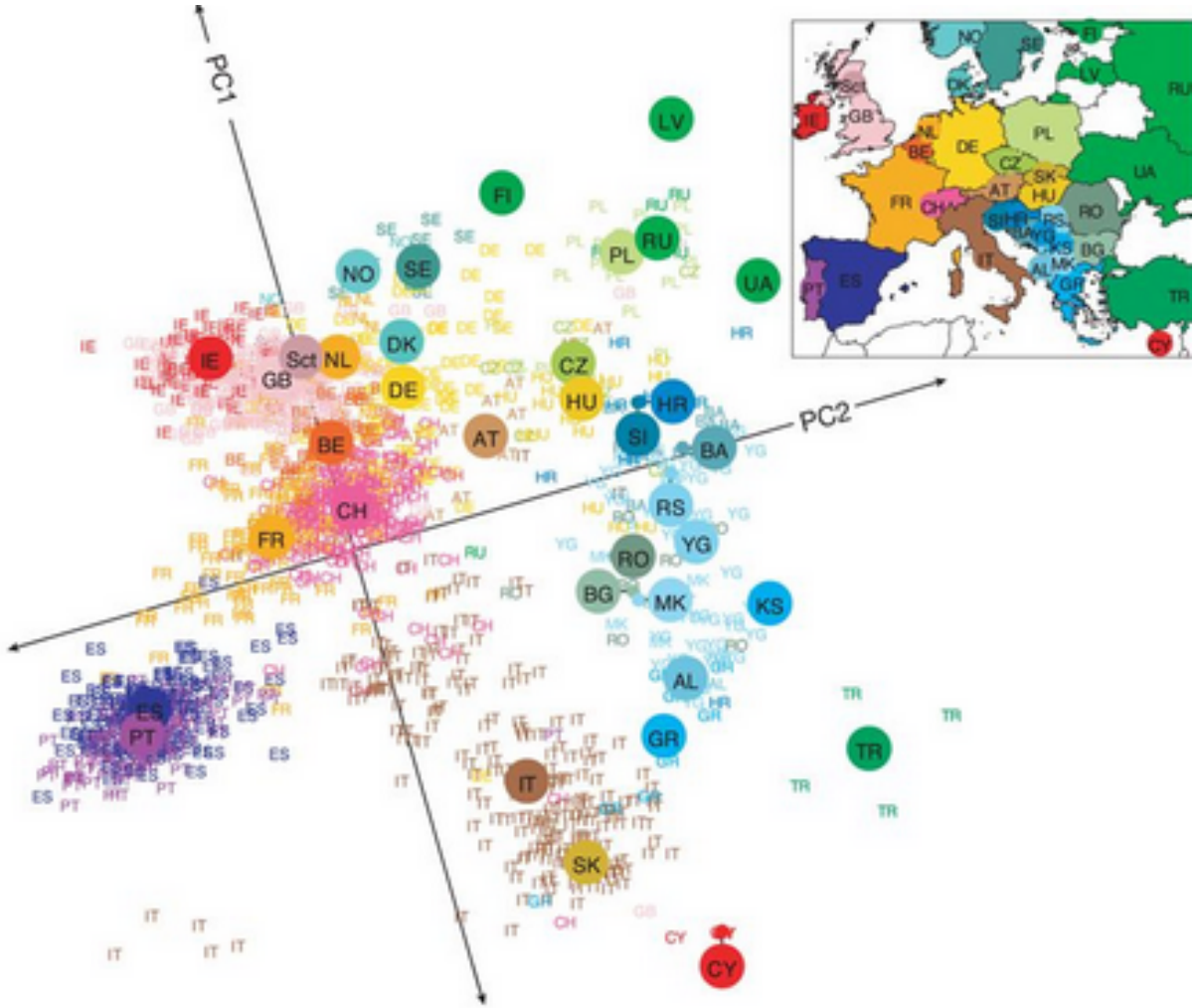
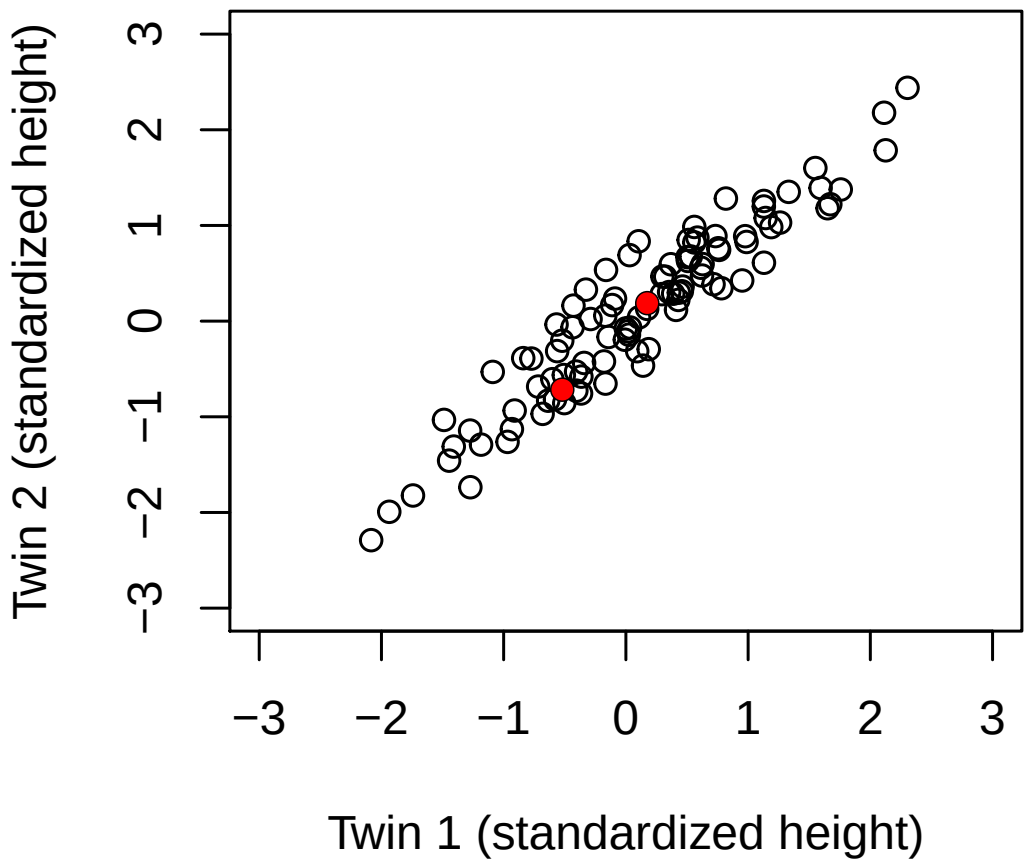


Figure 1: Population structure within Europe

Example: Twin heights

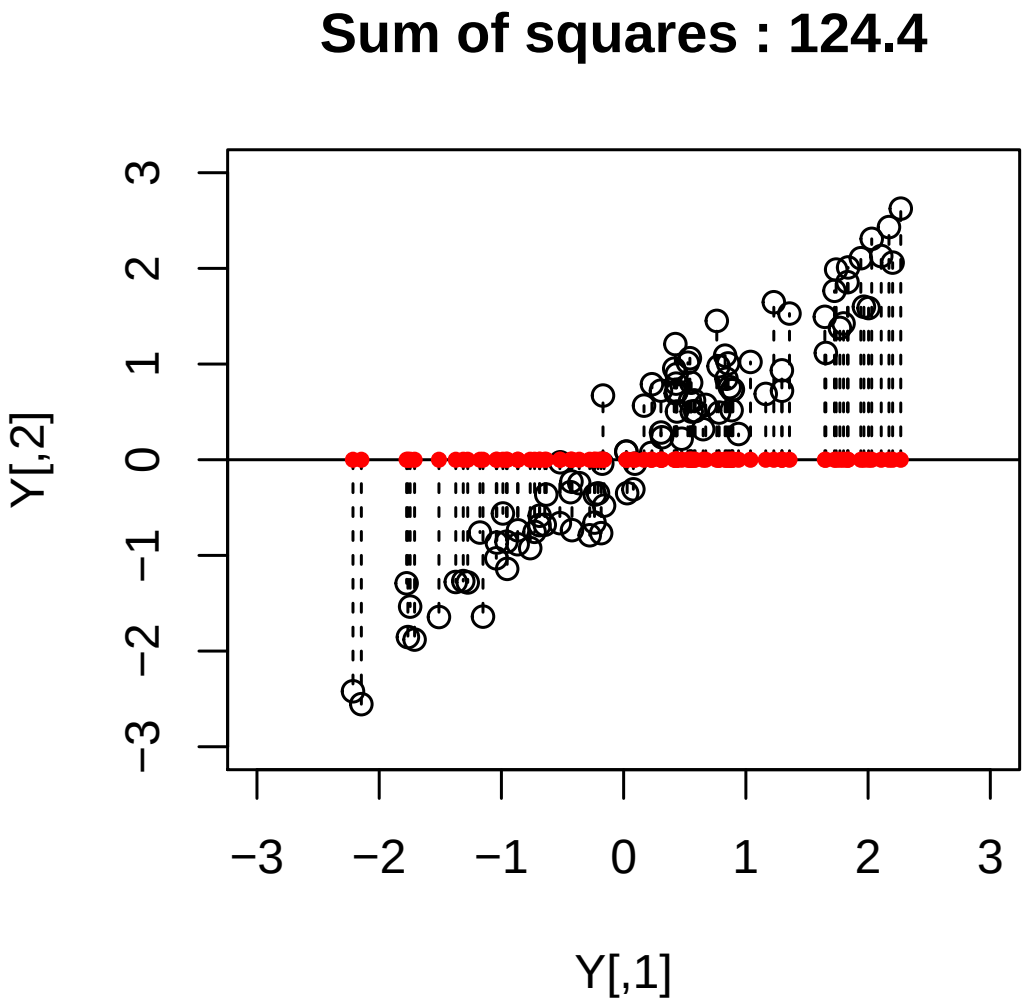
We started the motivation for dimension reduction with a simulated example and showed a rotation that is very much related to PCA.



Here we explain specifically what are the principal components (PCs).

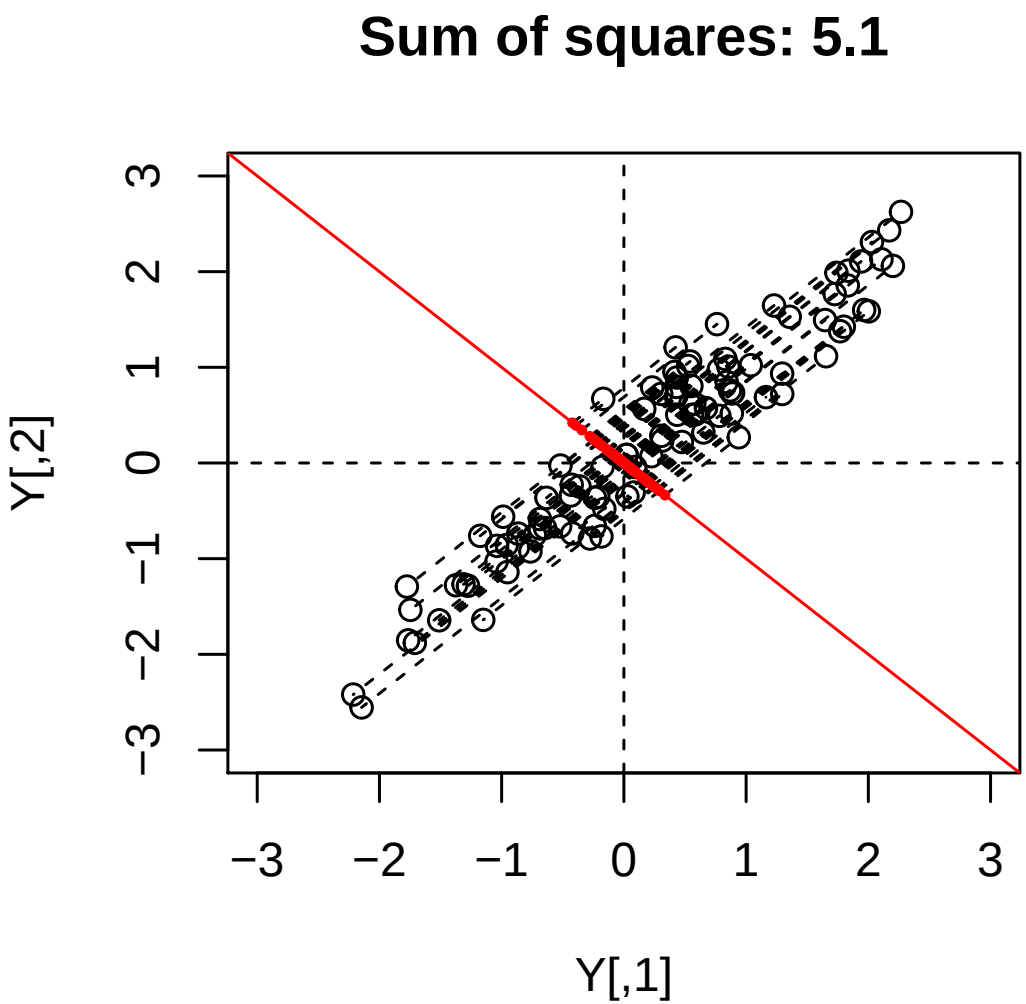
Let $\mathbf{Y}$ be $N \times 2$ matrix representing our data (N individuals 2 variables). Suppose we are given the task of finding a 1×2 vector $\mathbf{v}_1$ such that $\mathbf{v}_1^\top \mathbf{v}_1 = 1$ and it maximizes $(\mathbf{Y}\mathbf{v}_1^\top)^\top(\mathbf{Y}\mathbf{v}_1^\top)$. This can be viewed as a projection of each sample or column of $\mathbf{Y}$ into the subspace spanned by $\mathbf{v}_1$. So we are looking for a transformation in which the coordinates show high variability.

Let's try $\mathbf{v} = (1, 0)$. This projection simply gives us the height of twin 1 shown in orange below. The sum of squares is shown in the title.



Can we find a direction with higher variability? How about:

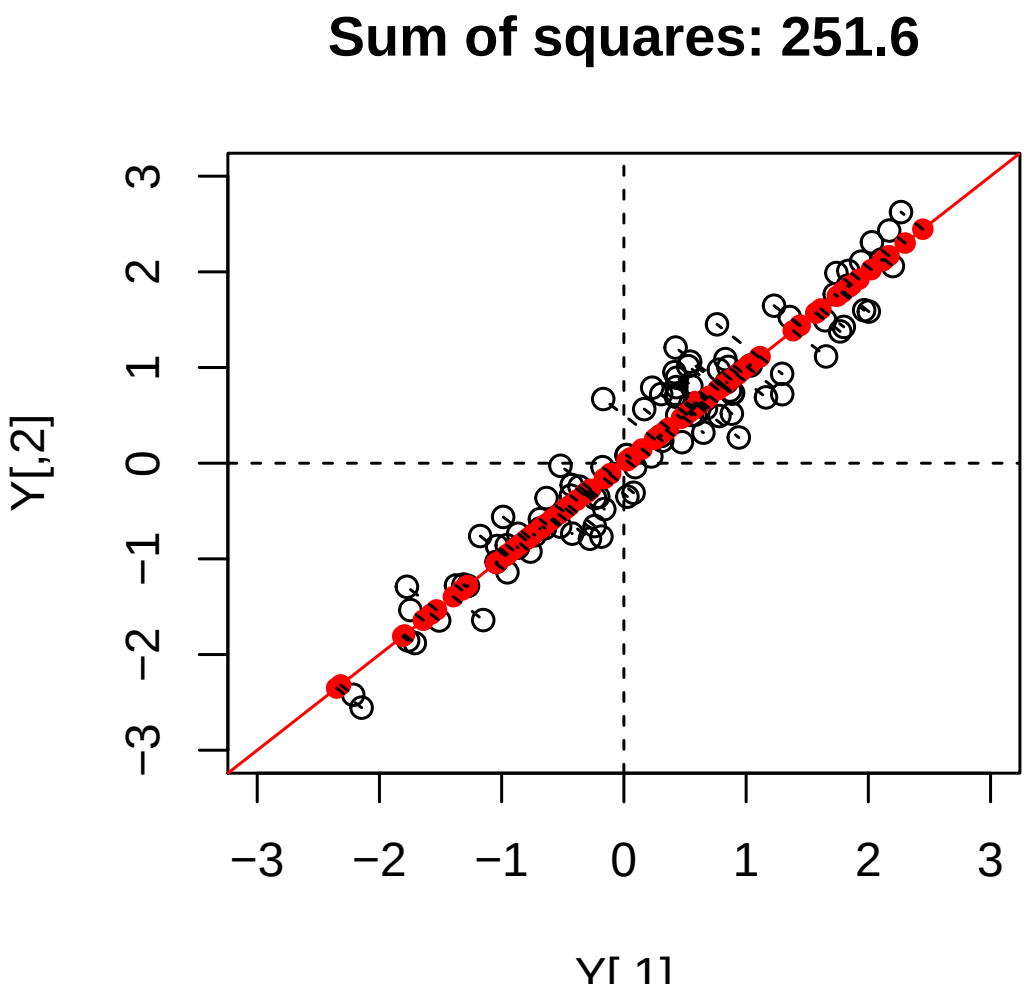
$\mathbf{v} = (1, -1)$? This does not satisfy $\mathbf{v}^\top \mathbf{v} = 1$ so let's instead try $\mathbf{v} = (1/\sqrt{2}, -1/\sqrt{2})$



This relates to the difference between twins, which we know is small. The sum of squares confirms this.

Finally, let's try:

$\mathbf{v} = (1/\sqrt{2}, 1/\sqrt{2})$



This is a re-scaled average height, which has higher sum of squares. There is a mathematical procedure for determining which **v** maximizes the sum of squares and the SVD provides it for us.

```
ss <- svd(Y)
v <- ss$v[,1]
v
```

```
[1] -0.6956108 -0.7184188
```

```
1/sqrt(2)
```

```
[1] 0.7071068
```

The sum of squares is:

```
w <- Y%*%v
crossprod(w) # t(w)%*%w
```

```
         [,1]
[1,] 251.6766
```

The principal components

The orthogonal vector that maximizes the sum of squares:

$$(\mathbf{Y}\mathbf{v}_1^\top)^\top(\mathbf{Y}\mathbf{v}_1^\top)$$

$\mathbf{Y}\mathbf{v}_1^\top$ is referred to as the first PC. The *weights* $\mathbf{v}$ used to obtain this PC are referred to as the *loadings*. Using the language of rotations, it is also referred to as the *direction* of the first PC, which are the new coordinates.

To obtain the second PC, we repeat the exercise above, but for the residuals:

$$\mathbf{r} = \mathbf{Y} - \mathbf{v}_1 \mathbf{Y} \mathbf{v}_1^\top$$

The second PC is the vector with the following properties:

$$\mathbf{v}_2 \mathbf{v}_2^\top = 1$$

$$\mathbf{v}_1 \mathbf{v}_2^\top = 0$$

and maximizes $\mathbf{r}\mathbf{v}_2 (\mathbf{r}\mathbf{v}_2)^\top$.

When Y is $N \times m$ we repeat to find 3rd, 4th, ..., m-th PCs.

prcomp

We have shown how to obtain PCs using the SVD. However, R has a function specifically designed to find the principal components. In this case, the data is centered by default. The following function:

```
pc <- prcomp(Y, center = FALSE)
```

produces the same results as the SVD up to arbitrary sign flips:

```
s <- svd( t(Y) )

# First axis
max(abs(pc$x[,1]) - abs(s$d[1]*s$v[,1]))
```

```
[1] 4.440892e-16
```

```
# Second axis
max(abs(pc$x[,2]) - abs(s$d[2]*s$v[,2]))
```

```
[1] 2.498002e-16
```

The loadings can be found this way:

```
pc$rotation
```

```
           PC1        PC2
[1,] 0.7078204  0.7063925
[2,] 0.7063925 -0.7078204
```

which are equivalent (up to a sign flip) to:

```
s$u
```

```
           [,1]       [,2]
[1,] -0.7078204 -0.7063925
[2,] -0.7063925  0.7078204
```

The equivalent of the variance explained is included in:

```
pc$sdev
```

```
[1] 1.263543 0.214353
```

```
s$d
```

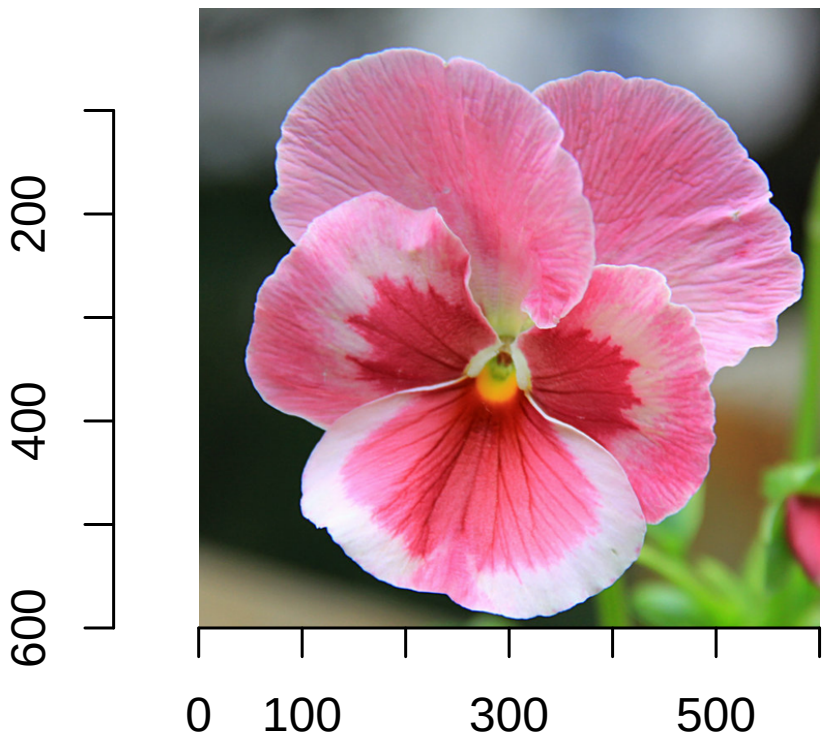
```
[1] 12.572096  2.132786
```

```
cumsum(pc$sdev)/sum(pc$sdev)
```

```
[1] 0.8549607 1.0000000
```

Example: data imaging

```
library(imager)
x <- load.image("figures/pansy.jpg")
plot(x)
```



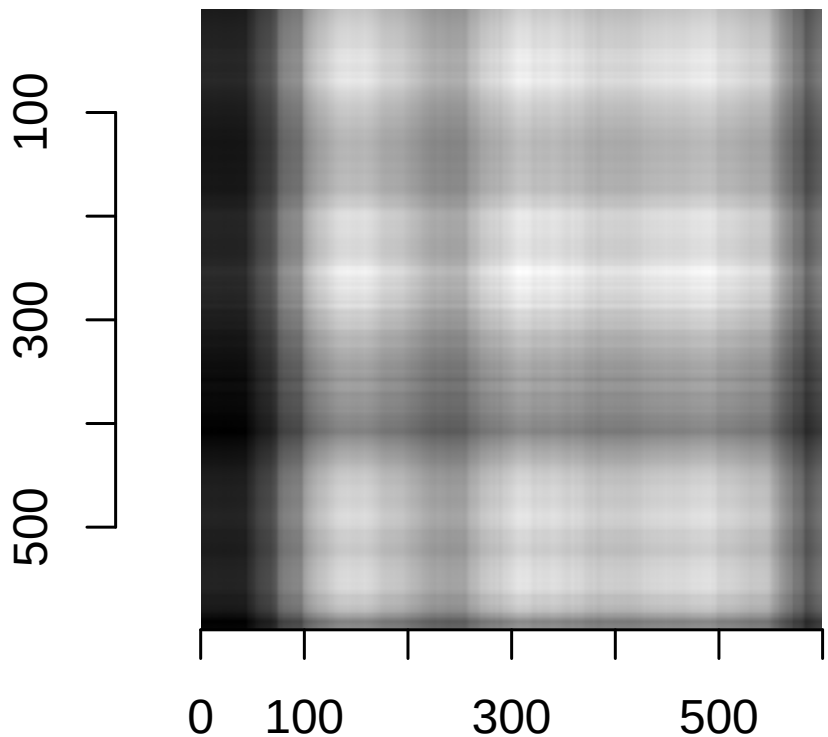
```
m <- grayscale(x)
plot(m)
```

Retrieving image

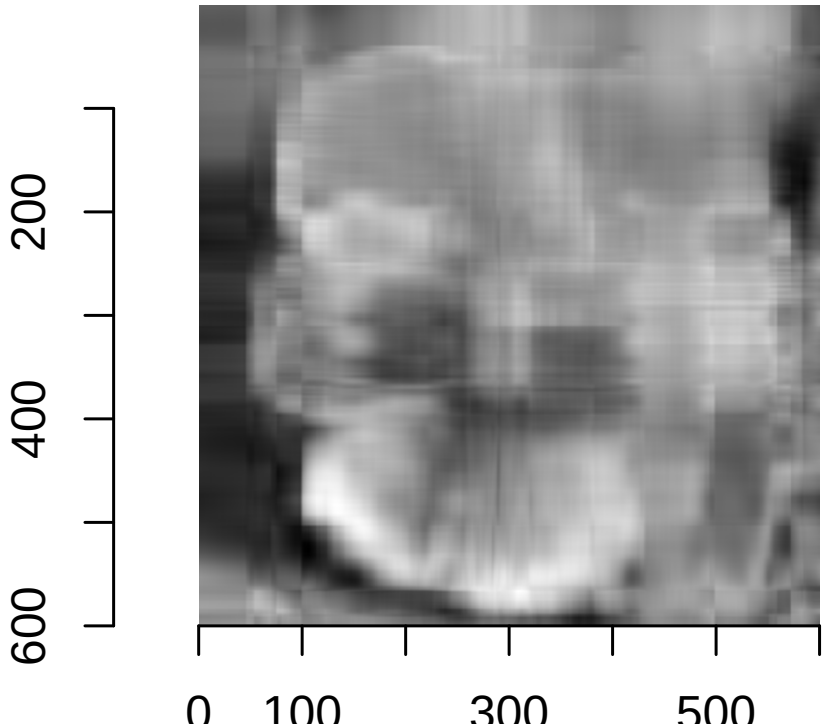
```
r.svd <- svd(m)
d <- diag(r.svd$d)
u <- r.svd$u
v <- r.svd$v

# first approximation
u1 <- as.matrix(u[-1, 1])
v1 <- as.matrix(v[-1, 1])
d1 <- as.matrix(d[1,1])
l1 <- u1 %*% d1 %*% t(v1)
l1g <- as.cimg(l1)
```

```
plot(l1g)
```



```r
# more depth
depth <- 8
us <- as.matrix(u[, 1:depth])
vs <- as.matrix(v[, 1:depth])
ds <- as.matrix(d[1:depth, 1:depth])
ls <- us %*% ds %*% t(vs)
lsg <- as.cimg(ls)
plot(lsg)
```



Large datasets (2-10 GB)

- PCA based on SVD scales as $O(\min(N^2 p, N p^2))$
- This makes it time-consuming to perform PCA in large datasets
- There are some solutions:
  - Randomized matrix algorithms[1], [2]
  - Partial or truncated SVD[3]

---

[1] Halko N, Martinsson PG, Shkolnisky Y, Tygert M (2011) An Algorithm for the Principal Component Analysis of Large Data Sets. SIAM Journal on Scientific Computing 33: 2580–2594.

[2] Halko N, Martinsson PG, Tropp JA (2011) Finding Structure with Randomness: Probabilistic Algorithms for Matrix Decompositions. SIAM Review 53: 217–288.

[3] Baglama, James, and Lothar Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. SIAM Journal on Scientific Computing 27.1 (2005): 19-42.

Exercise

- Find an R package that performs truncated SVD.
- Create a function (or write and R script) that performs PCA based on truncated SVD.
- Perform PCA analysis using this new R code and compare the results obtained using `prcomp` function on the data set `musk.txt` described here[4].
- Plot the molecules (e.g. observations/rows) in the first two axes and color each dot using the information given in the column `musk`.

[4] NOTE1: do not foget to remove last column; NOTE2: use 'microbenchmark' function to compare computation speed

`musk` dataset describes a set of 102 molecules (repeated measures, in total there are 476 observations) of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. The goal is to learn to predict whether new molecules will be musks or non-musks. In this task we only aim to see whether the two first principal components discriminate musks and non-musks molecules. Data columns represent:

- f_1 ... f_162: distance features measured in hundredths of Angstroms.
- f163: distance of the oxygen atom in the molecule to a designated point in 3-space. This is also called OXY-DIS.
- f164: OXY-X: X-displacement from the designated point.
- f165: OXY-Y: Y-displacement from the designated point.
- f166: OXY-Z: Z-displacement from the designated point.
- musk: 0:non-musk, 1:musk

Session info

```
sessionInfo()
```

```
R version 3.5.0 (2018-04-23)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

Matrix products: default

locale:
[1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
[3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
[5] LC_TIME=Spanish_Spain.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] MASS_7.3-50

loaded via a namespace (and not attached):
 [1] compiler_3.5.0   backports_1.1.2  magrittr_1.5     rprojroot_1.3-2
 [5] tools_3.5.0      htmltools_0.3.6  yaml_2.1.19      Rcpp_0.12.18
 [9] codetools_0.2-15 stringi_1.2.2    rmarkdown_1.9    knitr_1.20
[13] stringr_1.3.1    digest_0.6.15    evaluate_0.10.1
```