Intro to statistics for computational genomics

Davide Risso

Division of Biostatistics, School of Public Health
University of California, Berkeley

September 14, 2016

Introduction

Acknowledgements

Many of these slides are based or inspired by the material developed by Sandrine Dudoit for the course "Computational Statistics with Applications in Biology and Medicine" taught at UC Berkeley.

About me

- I am a postdoc in the Division of Biostatistics, University of California, Berkeley
- My research focuses on the development of statistical methods and software to address problems in biomedical and genomic applications.
- I am author and maintainer of four R/Bioconductor packages

Why statistics?

I keep saying that the sexy job in the next 10 years will be statisticians, and I'm not kidding.

Hal Varian, Chief Economist, Google (2009)

Why statistics?

The coming century is surely the century of data

David Donoho (2000)

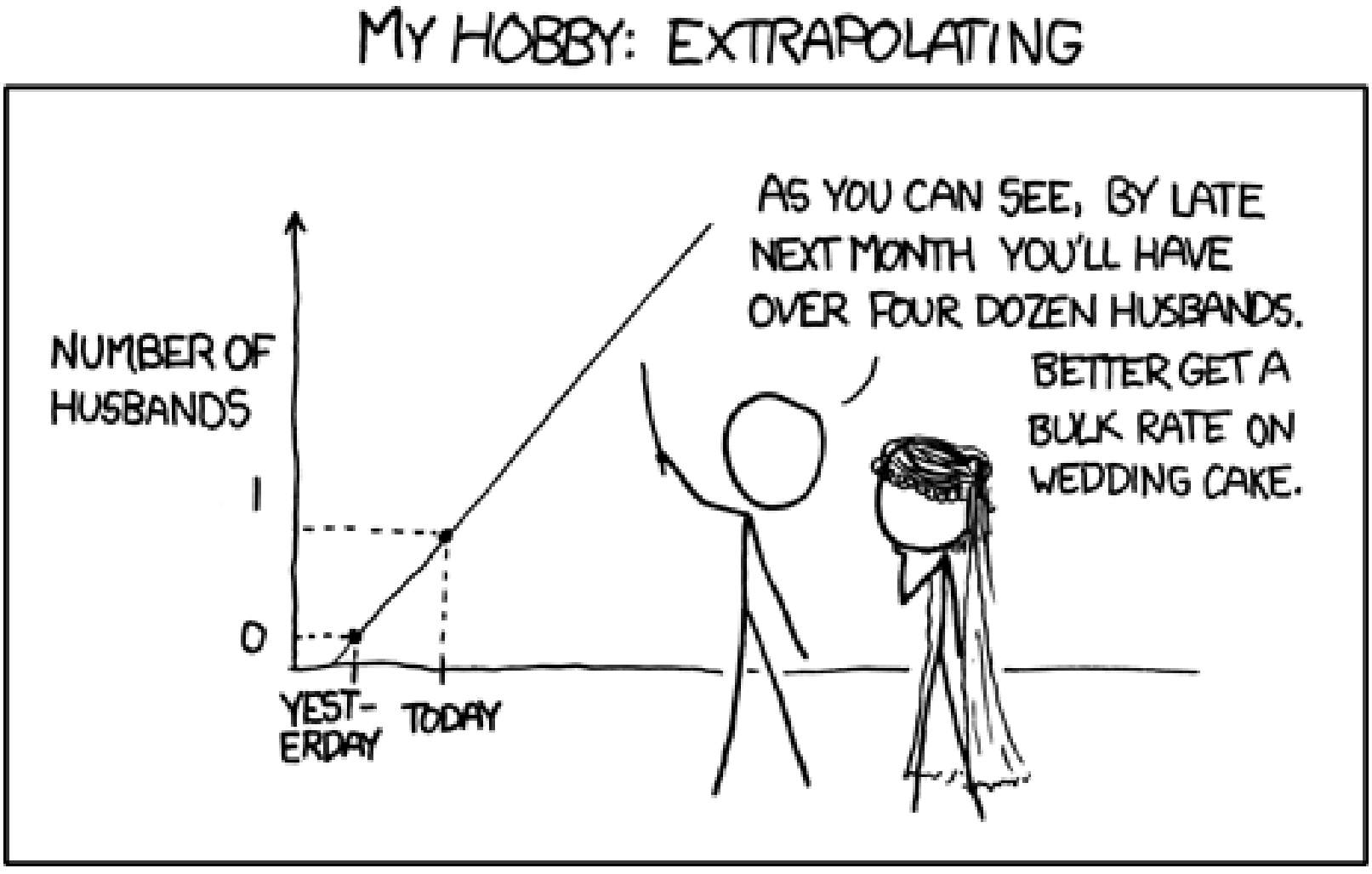
Statistical thinking will one day be as necessary for efficient citizenship as the ability to read or write

Attributed to H. G. Wells by Darrell Huff (1954)

Statistics is the grammar of science

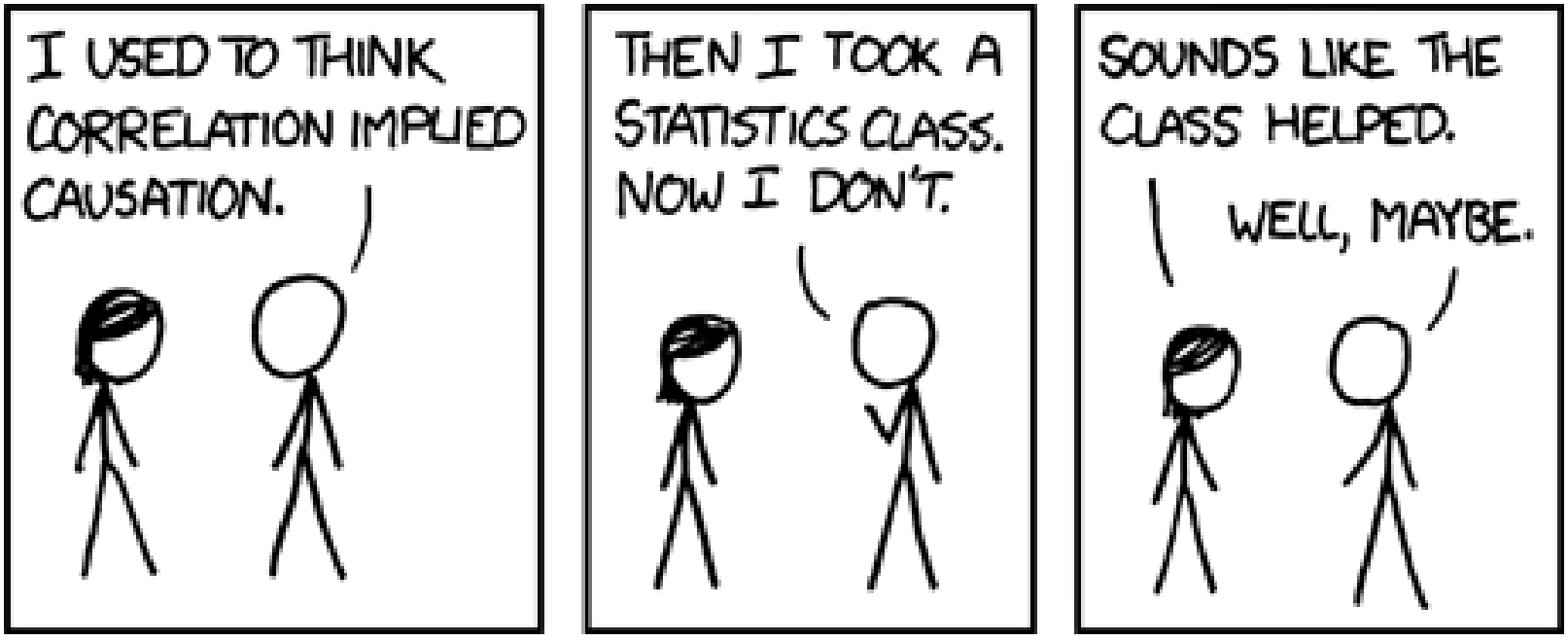
Karl Pearson (1892)

Why should *I* learn statistics?



http://xkcd.com/605/

Why should *I* learn statistics?



http://xkcd.com/552/

Why should *I* learn statistics?

Those who ignore Statistics are condemned to reinvent it

Attributed to Bradley Efron by Jerome H. Friedman (2001)

To call in the statistician after the experiment is done may be no more than asking him to perform a post-mortem examination: he may be able to say what the experiment died of.

R. A. Fisher (1938)

Why should *I* learn statistics?

Expression genetic data may not require a new breed of scientist, but they probably will require a new breed of collaboration. The focus of the computational biologist will need to change from the development of tools that answer specific questions to the development of general tools that enable biologists to carry out their own investigations—to explore, visualize and find biological signals in complex data.

Karl W. Broman (2005 Nat Genet 37: 209-210)

What is statistics?

Many views. . .

- We muddle through life making *choices based on incomplete information*. (Gonick and Smith, 1993).
- What makes Statistics unique is its *ability to quantify uncertainty*, to make it precise. This allows statisticians to make categorical statements, with complete assurance — about their level of uncertainty. (Gonick and Smith, 1993).
- Statistics is the *art of making numerical conjectures about puzzling questions*. (Freedman et al., 1978).
- The objective of statistics is to make inferences (predictions, decisions) about a population based on information contained in a sample. (Mendenhall, 1987).

What is statistics?

- *Descriptive statistics:* Summarize the data to highlight *trends* and discover *hidden patterns*.
- *Inference:* Make conclusions about a *population* based on information contained in a *sample*.

An example dataset: ALL/AML microarray data

Study of gene expression in two types of acute leukemias, acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML) (Golub et al. 1999).

- Affymetrix high-density oligonucleotide chips (Hu6800 chip): 7,129 different probe-sets.
- 47 cases of ALL (38 B-cell ALL and 9 T-cell ALL) and 25 cases of AML. Training set of 27 ALL and 11 AML.
- R datasets: `Golub_Train`, `Golub_Test`, and `Golub_Merge` in the Bioconductor package `golubEsets`.
- Data already pre-processed (image quantitation, normalization): measured on 16-bit scale, take log base 2.

```
source("https://bioconductor.org/biocLite.R")
biocLite("golubEsets")
```

An example dataset: ALL/AML microarray data

```
library(golubEsets)
library(magrittr)

data(Golub_Train)
golub<-exprs(Golub_Train)
golub[golub<100]<-100
golub[golub>16000]<-16000
golub <- log2(golub)
paste(Golub_Train$T.B.cell, Golub_Train$ALL.AML) %>%
  sub("NA", "", .) %>% as.factor -> pheno
head(golub[,1:3])
```

```
##                          1        2        3
## AFFX-BioB-5_at    6.643856 6.643856 6.643856
## AFFX-BioB-M_at    6.643856 6.643856 6.643856
## AFFX-BioB-3_at    6.643856 6.643856 6.643856
## AFFX-BioC-5_at    6.643856 8.144658 8.271463
## AFFX-BioC-3_at    6.643856 6.643856 6.643856
## AFFX-BioDn-5_at   6.643856 6.643856 6.643856
```

Numerical and graphical summaries

Definitions: Data and variables

- The *data* consist of one or more *variables* measured/recorded on observational units in a *population* or *sample* of interest.
- The term *variable* refers to characteristics that differ among observational units.
- E.g., expression of genes (variables) in patients (observational units)

Definitions: Data and variables

Quantitative/Numerical variables

- *Continuous* (real numbers): any value corresponding to the points in an interval (e.g., height, cholesterol level).
- *Discrete* (integers): countable number of values (e.g., T-cell counts).

Qualitative/Categorical variables

- *Nominal*: names or labels, no natural order (e.g, sex, eye color).
- *Ordinal*: ordered categories, no natural numerical scale (e.g., tumor grade).

Numerical and Graphical Summaries of Data

The goal is to *provide numerical and graphical descriptions of data* to

- summarize the main features of the data;
- uncover unusual features of the data;
- reveal and summarize relationships between one or more variables and/or observational units.

Useful for *exploratory data analysis* (EDA) (cf. Tukey):

- quality control
- overall impressions
- outlier detection
- validity of the assumptions of candidate models

and for *displaying and reporting the results* of a statistical analysis.

Summarizing the data

How can we get a sense of the values of the gene Zyxin ("ZYX") in this dataset?

```
ZYX <- golub["X95735_at",]
ZYX
```

```
##         1         2         3         4         5         6         7
##  8.219169  8.262095  8.271463  9.436712  9.477758  7.948367  6.643856
##         8         9        10        11        12        13        14
##  6.643856  6.643856  8.942515  6.643856  8.672425  9.499846  8.491853
##        15        16        17        18        19        20        21
##  9.873444  9.727920  9.854868  8.607330  9.264443  9.691744  6.643856
##        22        23        24        25        26        27        34
##  6.930737  6.643856  7.794416  9.451211  8.758223  6.643856 11.525031
##        35        36        37        38        28        29        30
## 10.036174 12.247631 11.350939 10.706496 11.118941 11.709084 12.602235
##        31        32        33
## 10.596190 11.686938 11.765700
```

Histograms

```
library(RColorBrewer)
colors <- brewer.pal(8, "Set2")
hist(ZYX, col=colors[1])
```

Histograms

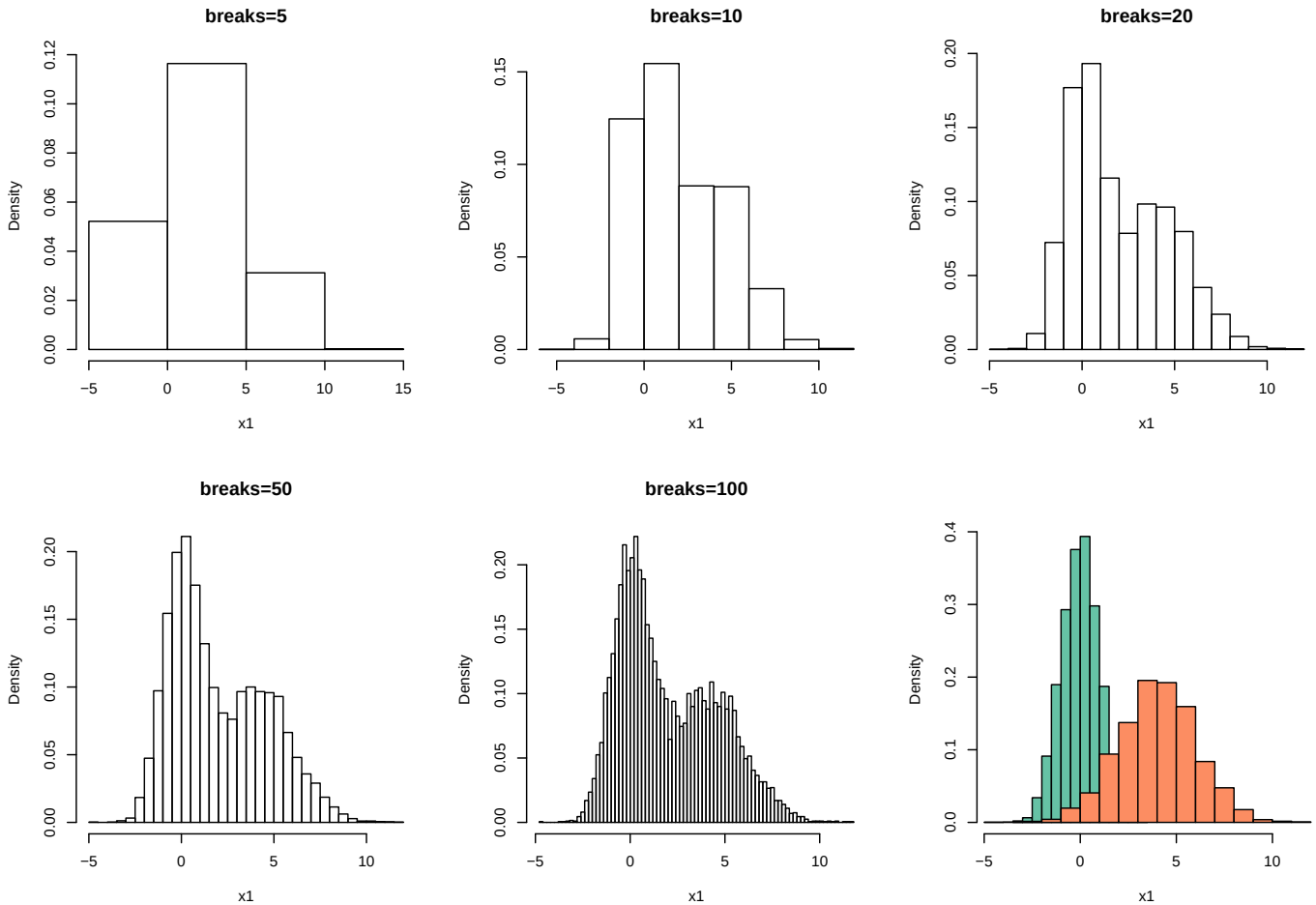
A histogram consists of a set of blocks or bins, where the area of each block represents the percentage of observations in the corresponding class interval.

The unit of the vertical axis is percent per unit of the horizontal axis.

The choice of the number of bins (argument `breaks`) greatly affects the plot.

Histograms

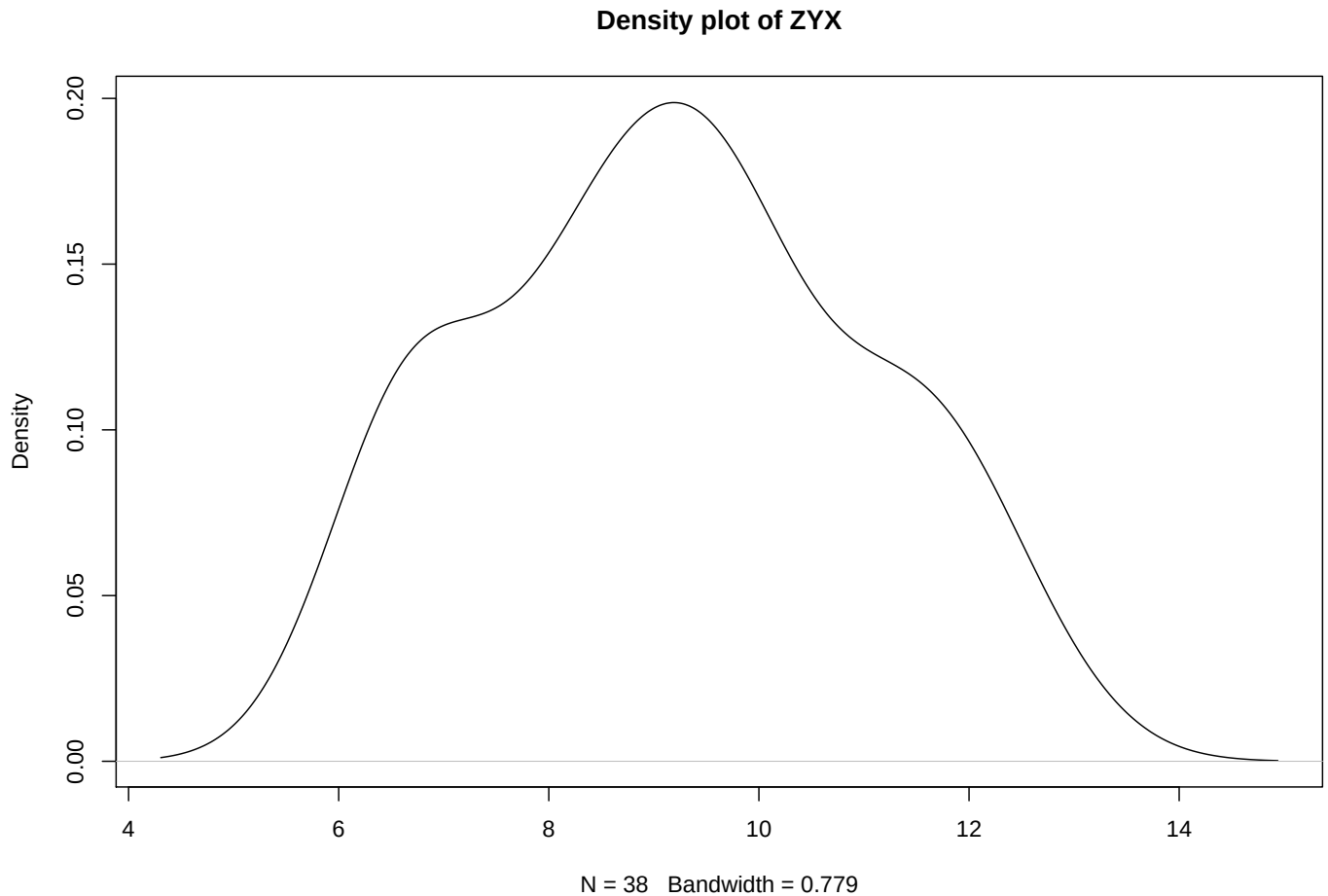
Same data with different number of bins.



Density plots

Smoothed versions of histograms, using *kernel density estimators*.

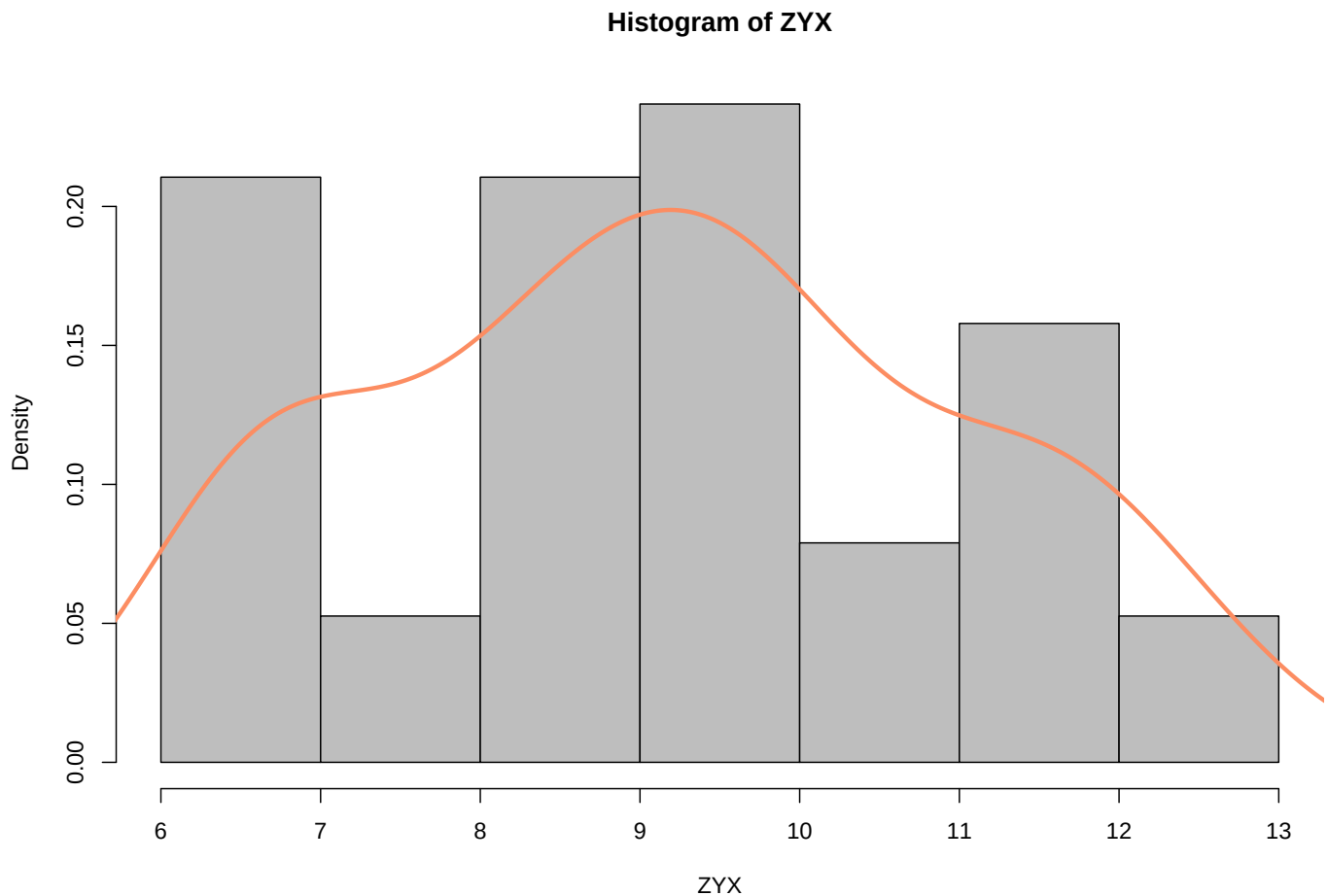
```
plot(density(ZYX), main="Density plot of ZYX")
```



Density plots

Smoothed versions of histograms, using *kernel density estimators*.

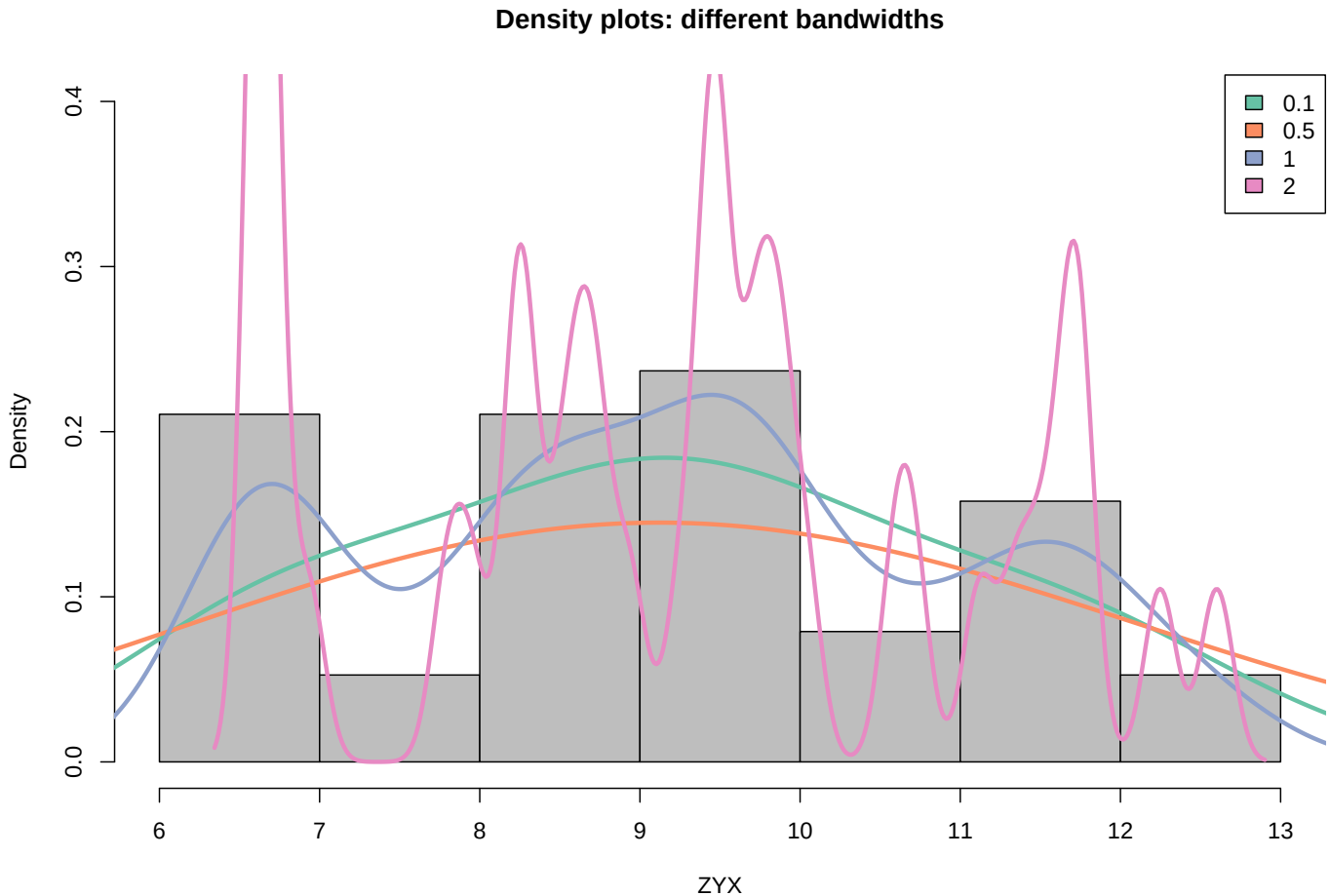
```
hist(ZYX, col="gray", probability=TRUE)
lines(density(ZYX), lwd=3, col=colors[2])
```



Density plots

- The choice of bandwidth is very important and can have a large impact on the density estimator.
- The larger the bandwidth, the smoother the estimator.
- This is an example of a bias-variance trade-off; as the bandwidth increases, variance decreases but bias increases.

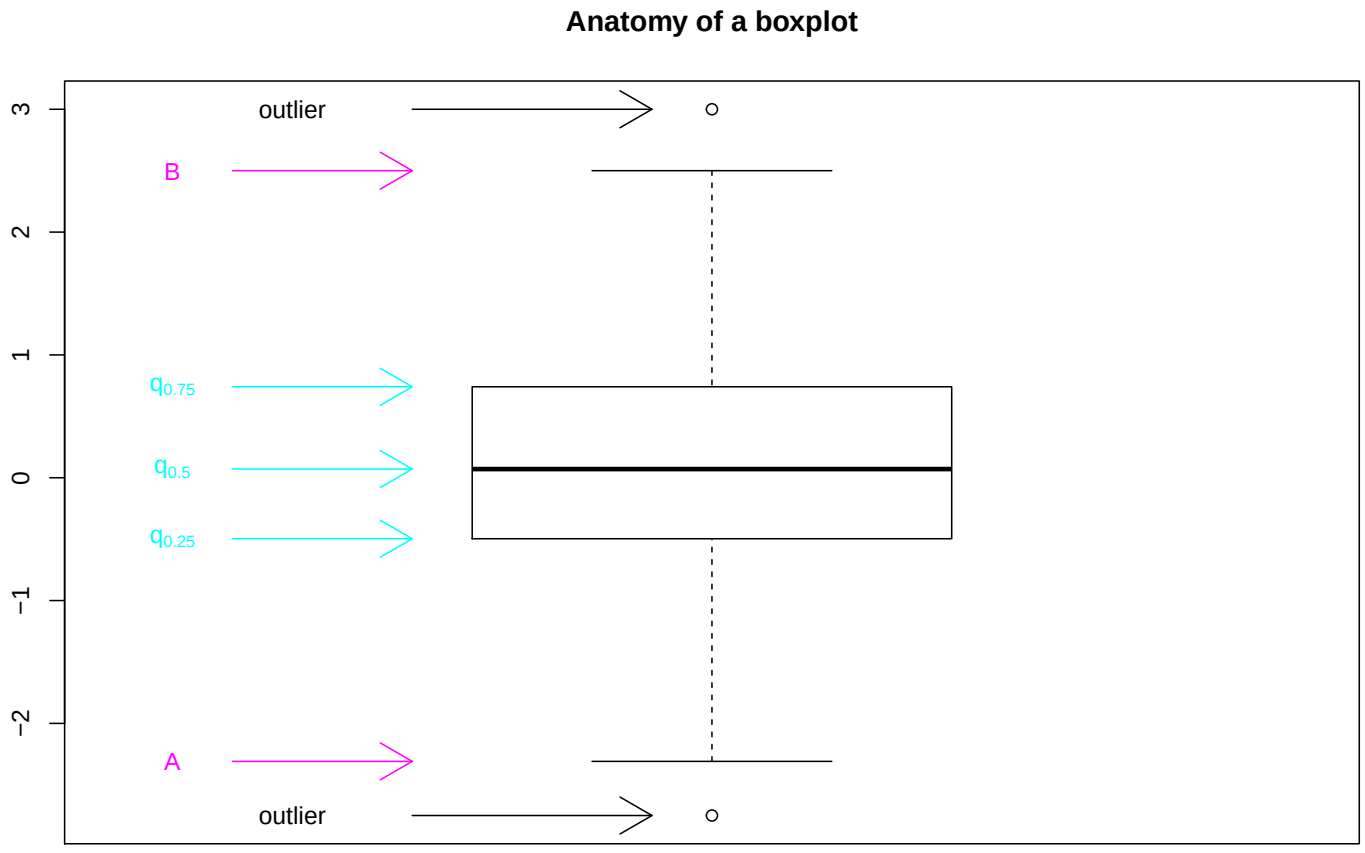
Density plots



Boxplots

- The *boxplot*, also called *box-and-whisker plot*, was proposed by Tukey (1977) as a simple graphical summary of the distribution of a variable.
- The summary consists of the median, the upper and lower quartiles, the range, and possibly individual extreme values.

Boxplots



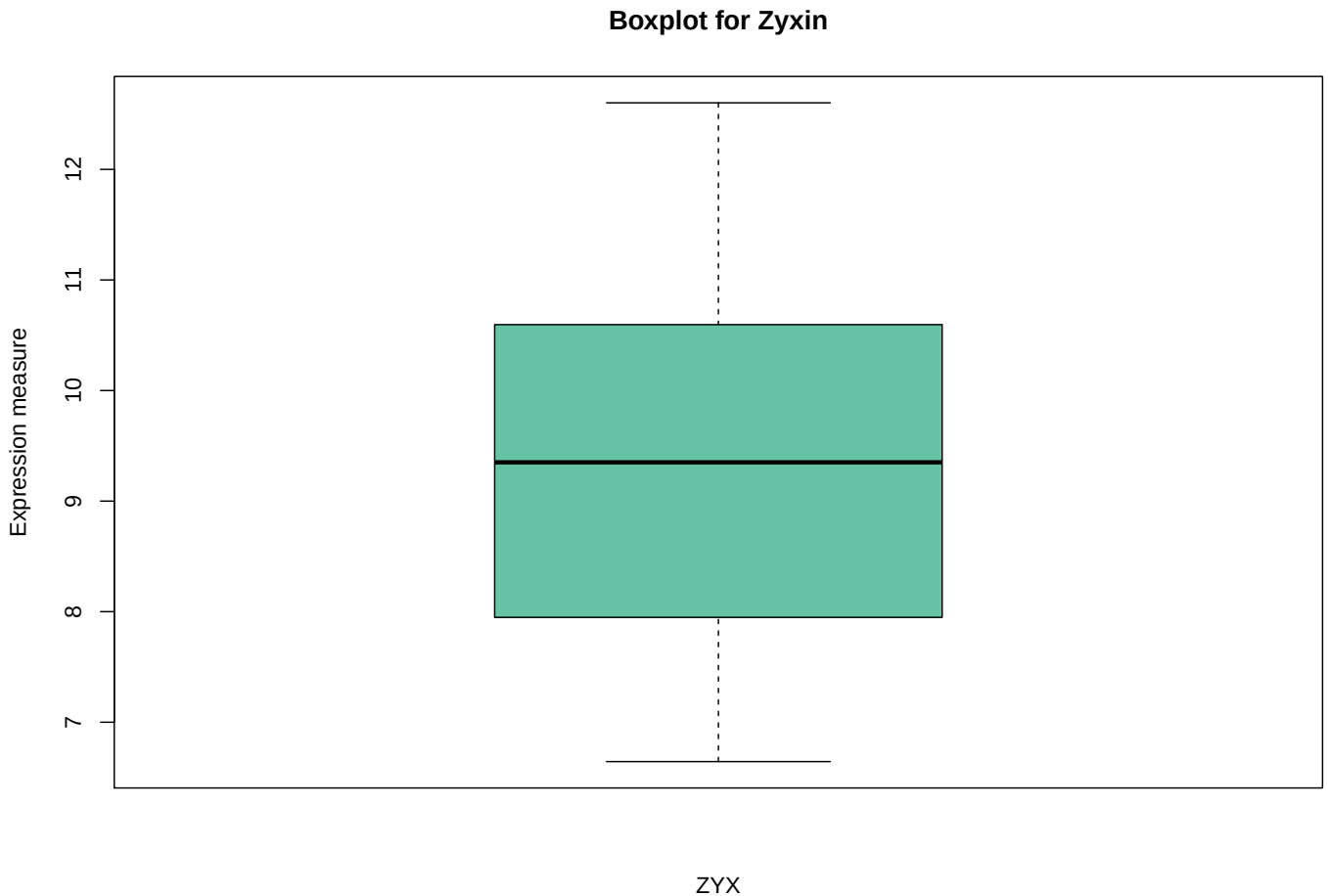
J. H. Bullard, K. D. Hansen, and M. Taub (Summer 2008). Statistics with R for Biologists: A Short Course.

Boxplot

- The line in the middle of the box represents the median, a robust measure of the center or location of the distribution.
- The upper and lower sides of the box are the upper and lower quartiles, respectively.
- The central box represents the inter-quartile range (IQR), a robust measure of the spread or scale of the distribution.
- Extreme values, more than 1.5 IQR above the upper quartile and below the lower quartile, are typically plotted individually.

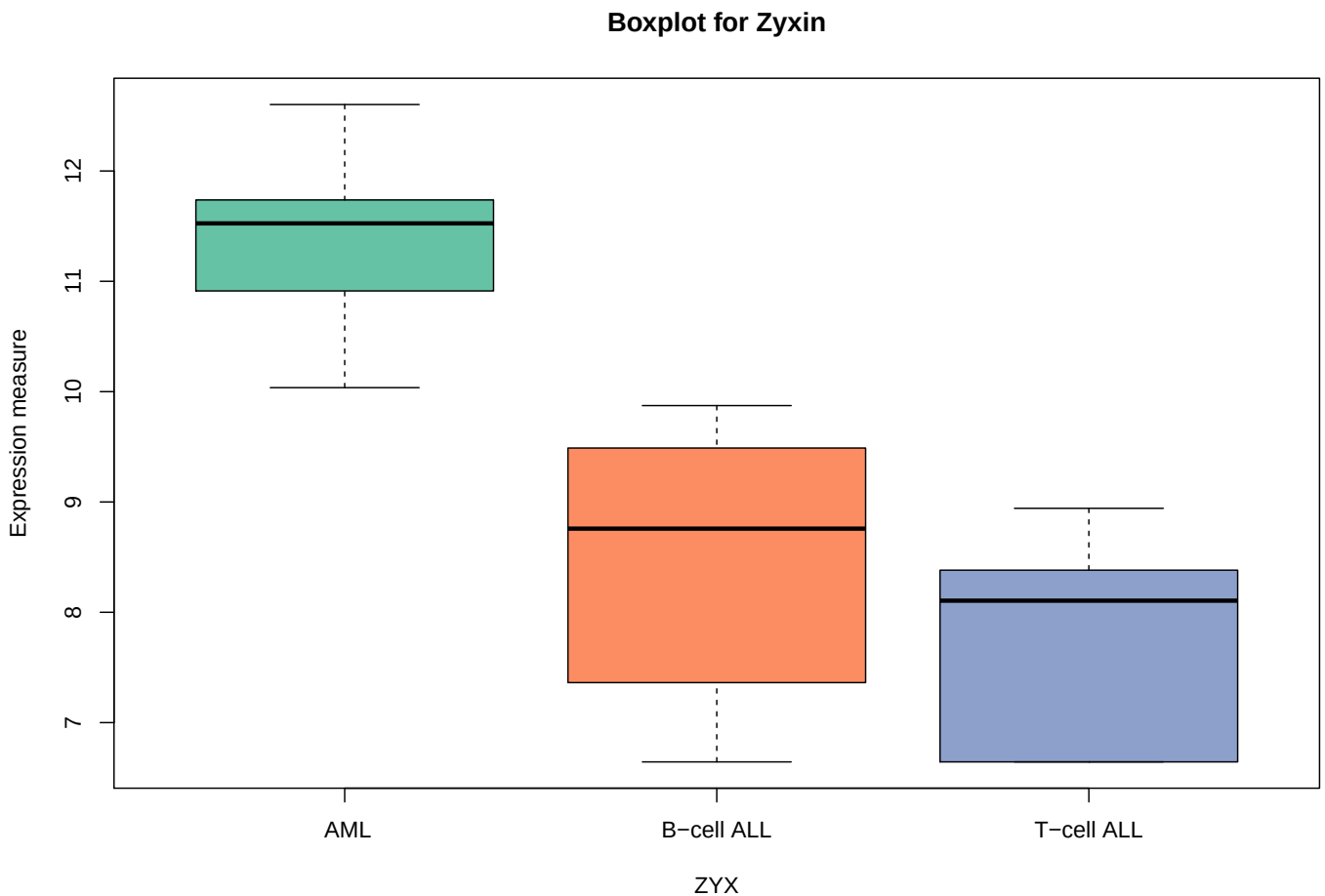
Boxplot

```
boxplot(ZYX, xlab="ZYX", ylab="Expression measure",
        col=colors[1], main="Boxplot for Zyxin")
```



Boxplot

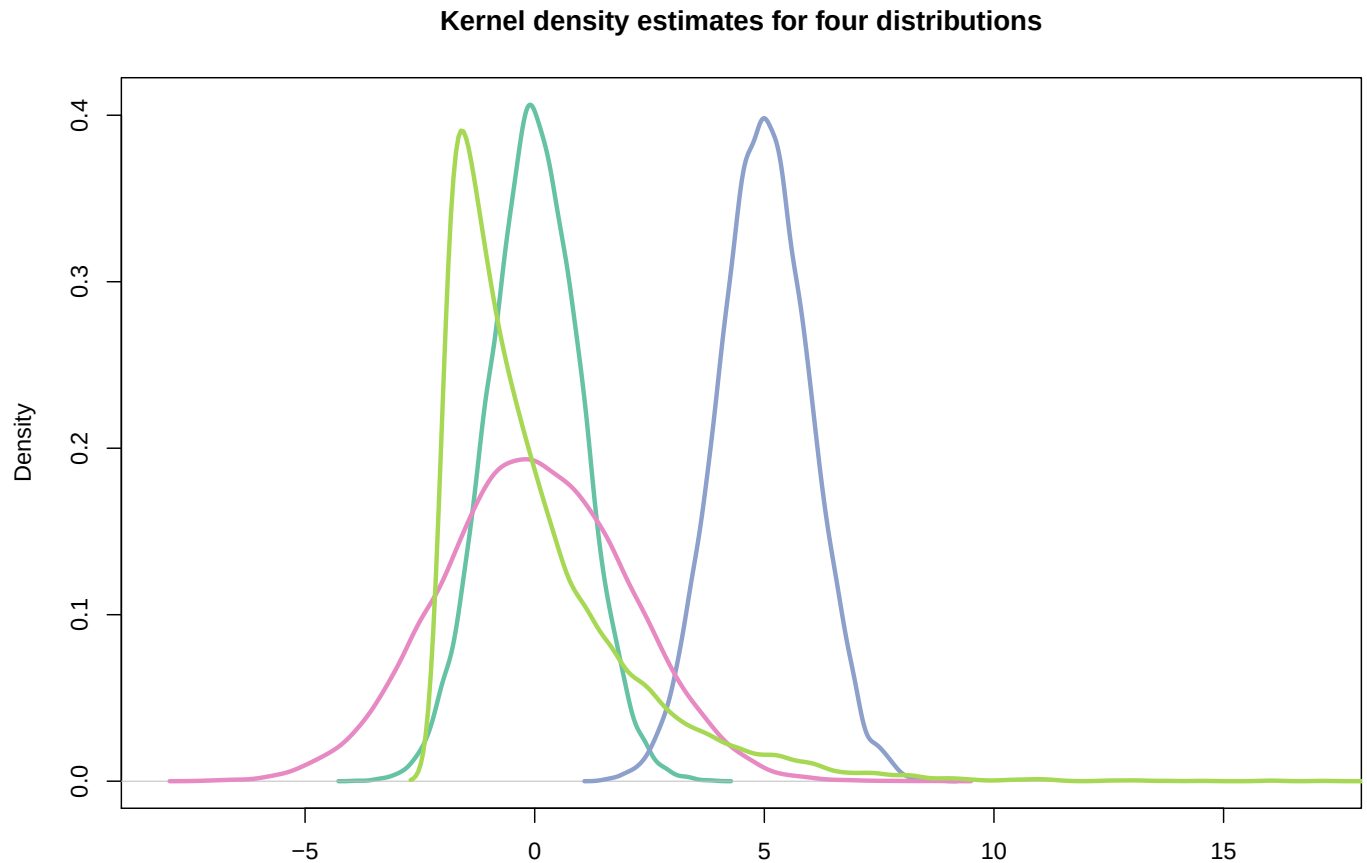
```
boxplot(ZYX~pheno, xlab="ZYX", ylab="Expression measure",
        col=colors, main="Boxplot for Zyxin")
```



Numerical summaries

What if we want to *quantify* the differences between the distributions?

As an example, consider the following four sets of 10,000 numbers.



Location, spread, range

Consider a list of n real numbers, $\{x_1, ..., x_n\}$.
The *mean* measures the *center* or *location* of a distribution,

$$\bar{x} \equiv \frac{\text{sum of values}}{\text{number of values}} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

The *standard deviation* (SD) measures the *spread* or *scale* of a distribution.

$$\begin{aligned} s_x &\equiv \sqrt{\text{mean of (deviations from the mean)}^2} \\ &= \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}. \end{aligned}$$

The *variance* is the square of the standard deviation.

For the Gaussian distribution, roughly 68% of the observations are within one SD of the mean and 95% within two SD.
R functions: `mean`, `sd`, `var`.

Robust alternatives

The *median* is another measure describing the center of a distribution.
It is more *robust* than the mean, in the sense that it is less affected by extreme values, i.e., by observations in the tails of the distribution.

Similarly, robust measures of *spread* are the *inter-quartile range* (IQR),

$$IQR \equiv \text{upper quartile} - \text{lower quartile},$$

and the *median absolute deviation* (MAD),

$$MAD \equiv \text{median of } |\text{deviations from median}|.$$

R functions: `median`, `mad`, and `IQR`.

Robustness

```
x <- c(rnorm(100, mean=0, sd=1), 1000)
mean(x)
```

```
## [1] 9.758488
```

```
median(x)
```

```
## [1] -0.04034829
```

```
sd(x)
```

```
## [1] 99.5224
```

```
mad(x)
```

```
## [1] 0.9916473
```

```
IQR(x)
```

```
## [1] 1.314821
```

Relation between two variables: correlation

The *correlation coefficient* is a measure of *linear association*. It is defined as the mean of the product of the observations in standard units,

$$r = \text{Cor}[x, y] \equiv \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right).$$

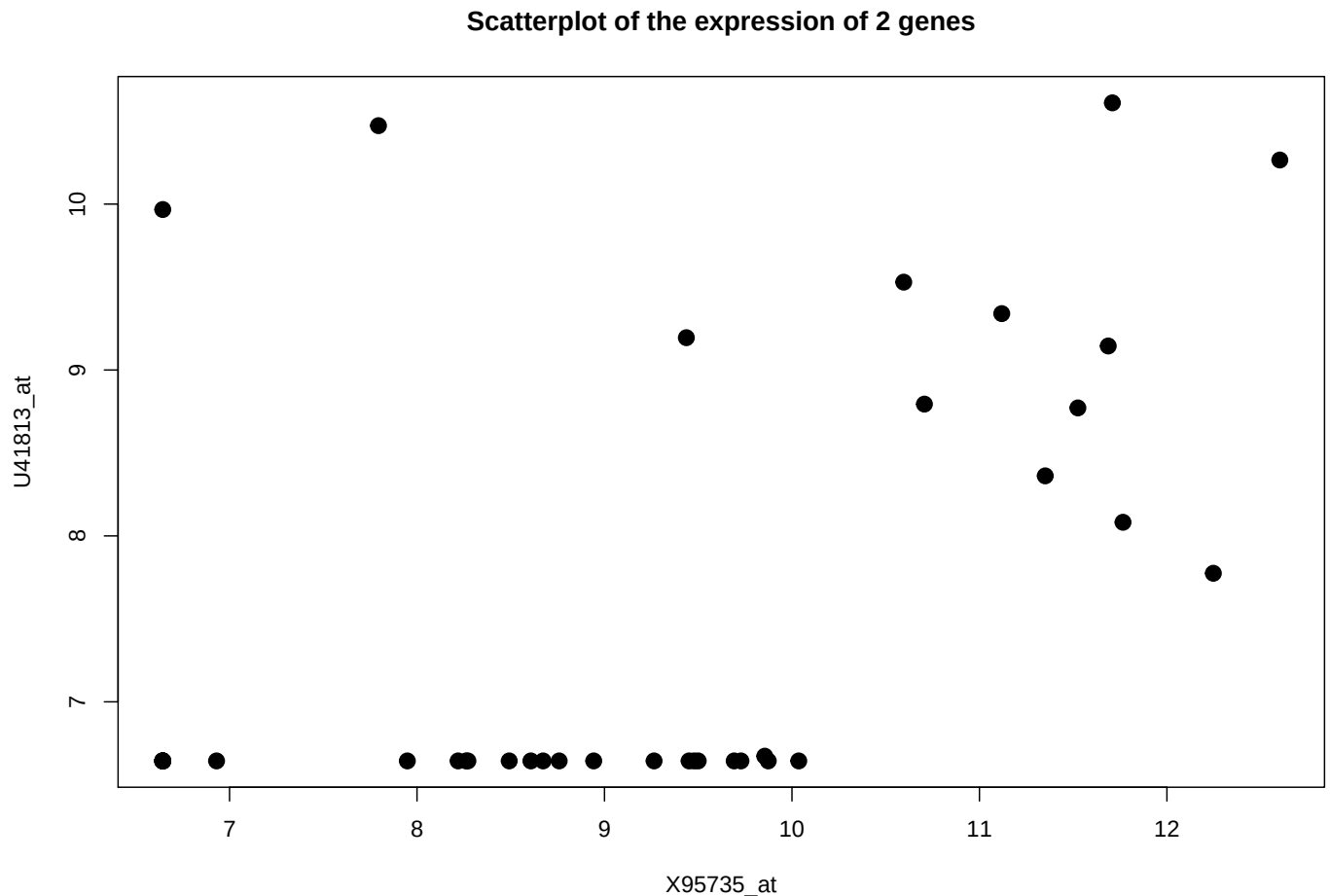
Correlation coefficients are always *between -1 and 1*.
A positive (negative) correlation means that the cloud of points in the scatterplot slopes up (down).

The correlation coefficient is *without units*.
It is not affected by: adding a constant to all the values of one variable; multiplying all the values of one variable by a positive constant.

Spearman's rank correlation coefficient is a robust measure of correlation, where ranks are used in place of the actual values of the x and y variables.

Scatterplot

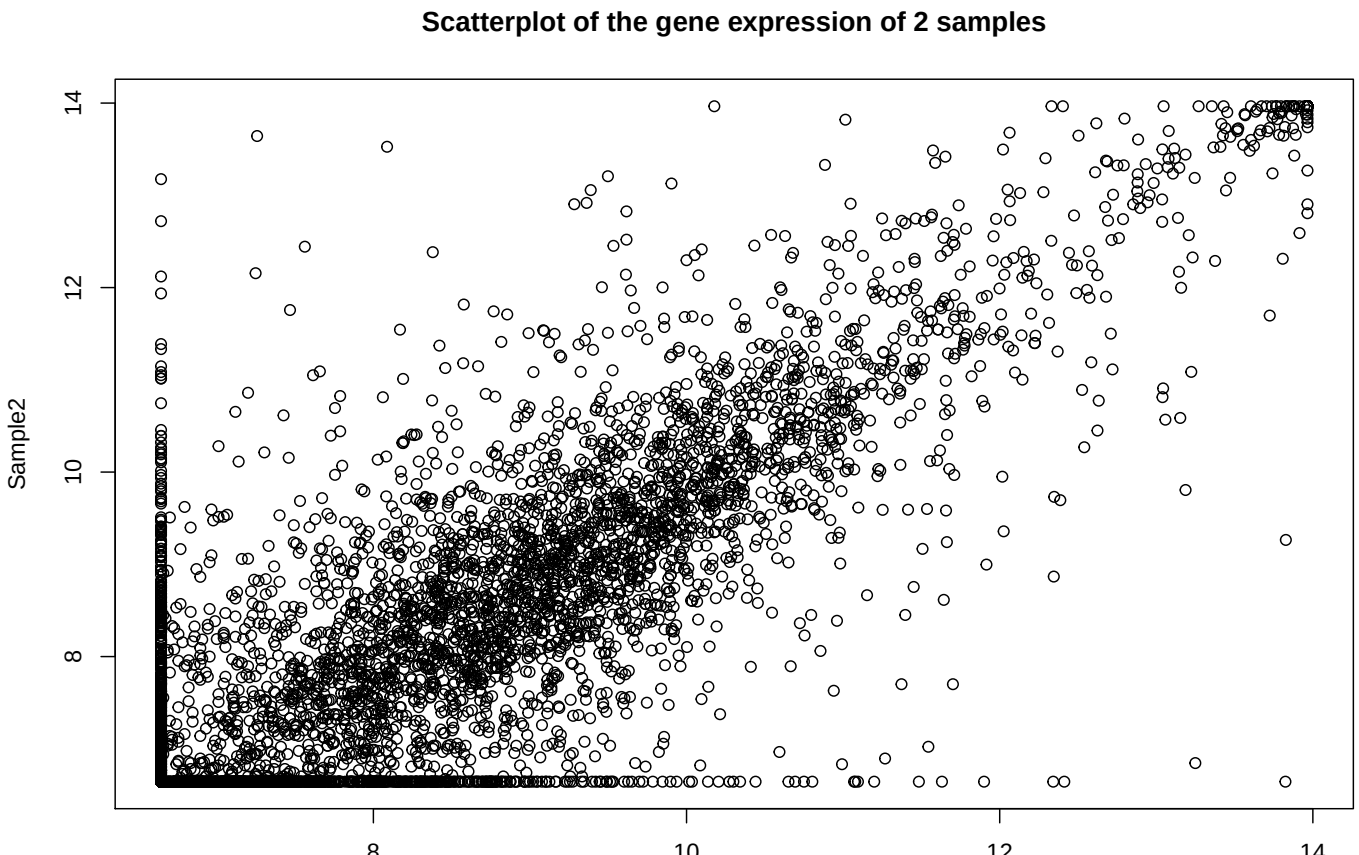
```
golub2 <- t(golub[c("X95735_at", "U41813_at"),])
plot(golub2, main="Scatterplot of the expression of 2 genes",
     pch=19, cex=1.5)
```



Smoothing

When there are many points, scatterplots are not very informative because of *overplotting*.

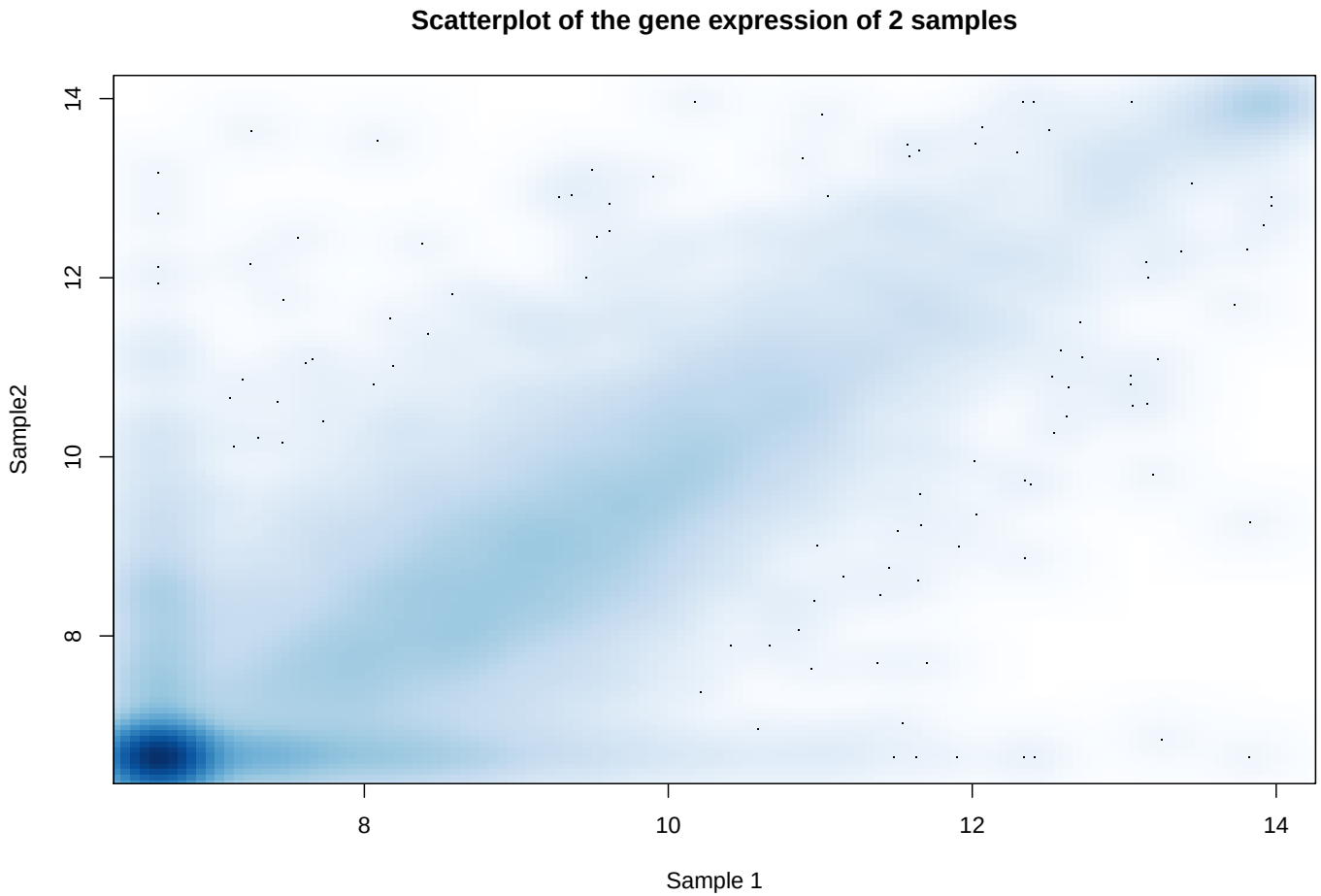
```
golub_samples <- golub[, 1:2]
plot(golub_samples, xlab="Sample 1", ylab="Sample2",
     main="Scatterplot of the gene expression of 2 samples"
```



Smoothing

One solution is *smoothing*.

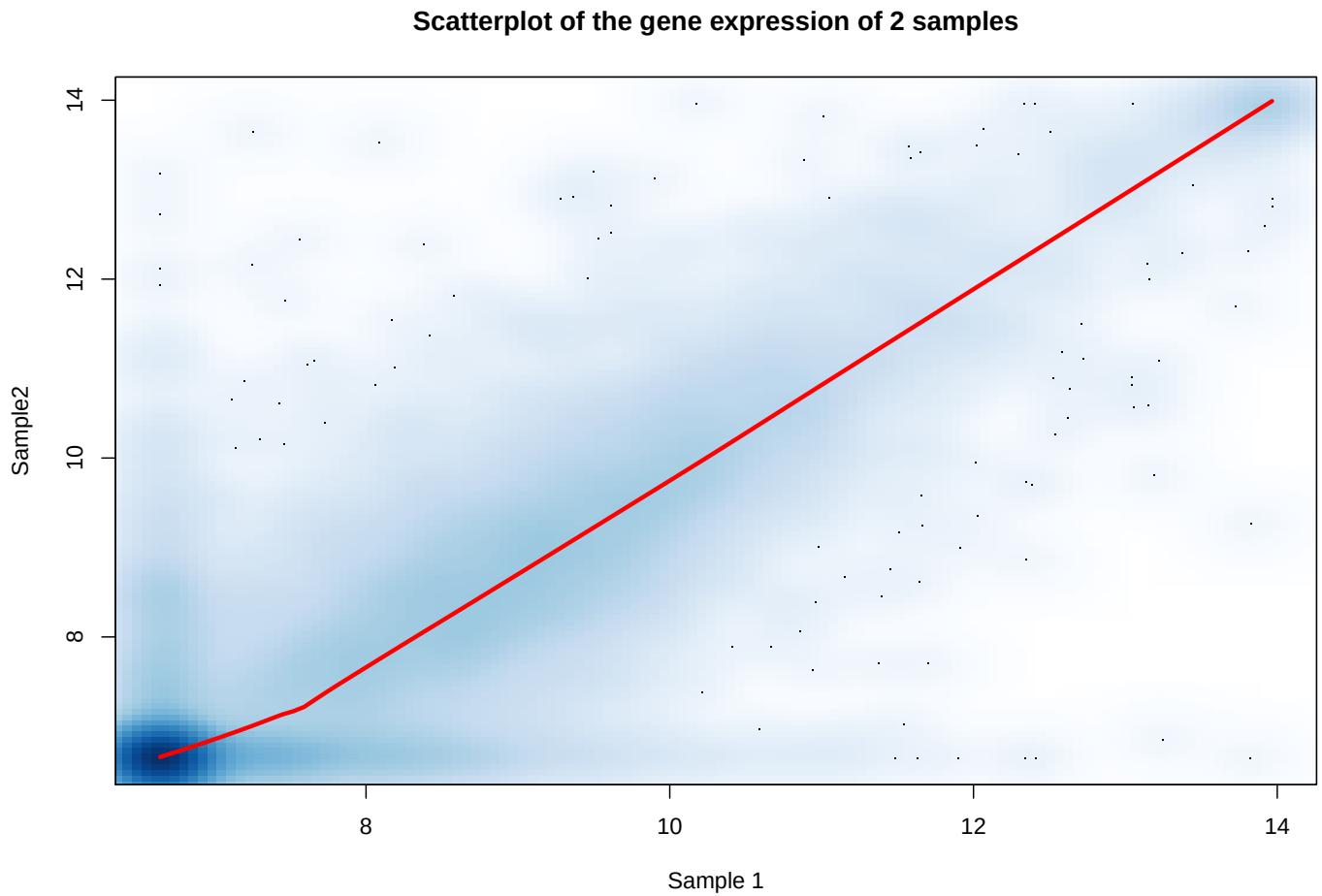
```
smoothScatter(golub_samples, xlab="Sample 1", ylab="Sample2
     main="Scatterplot of the gene expression of 2 samples"
```



Smoothing

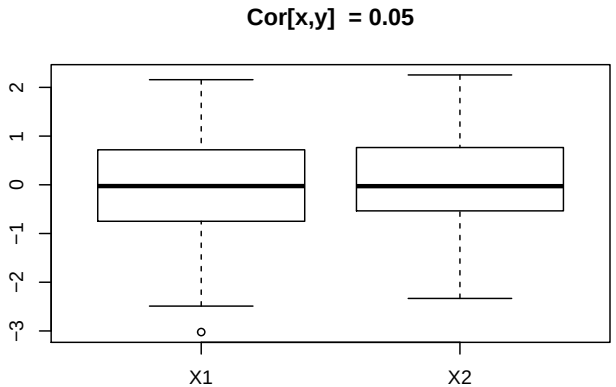
Adding a trend with *lowess regression* can reveal hidden patterns.

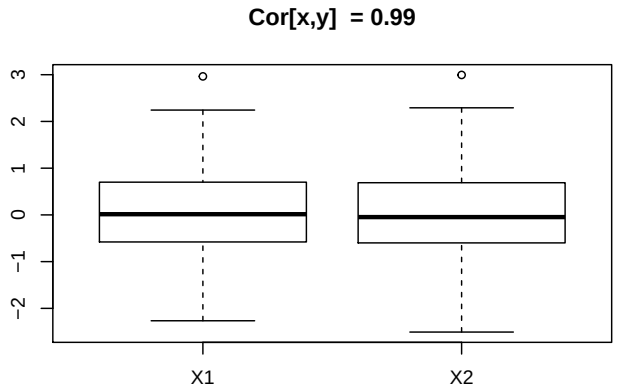
```
smoothScatter(golub_samples, xlab="Sample 1", ylab="Sample2
     main="Scatterplot of the gene expression of 2 samples"
lines(lowess(golub_samples), lwd=3, col=2)
```

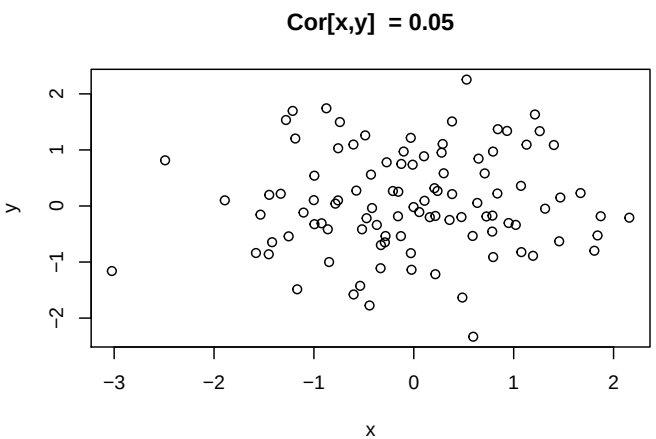


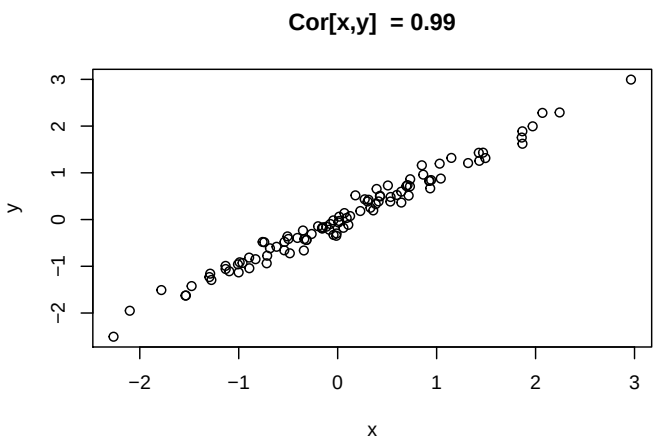
The importance of EDA

Different numerical and graphical summaries highlight different aspects of the data.

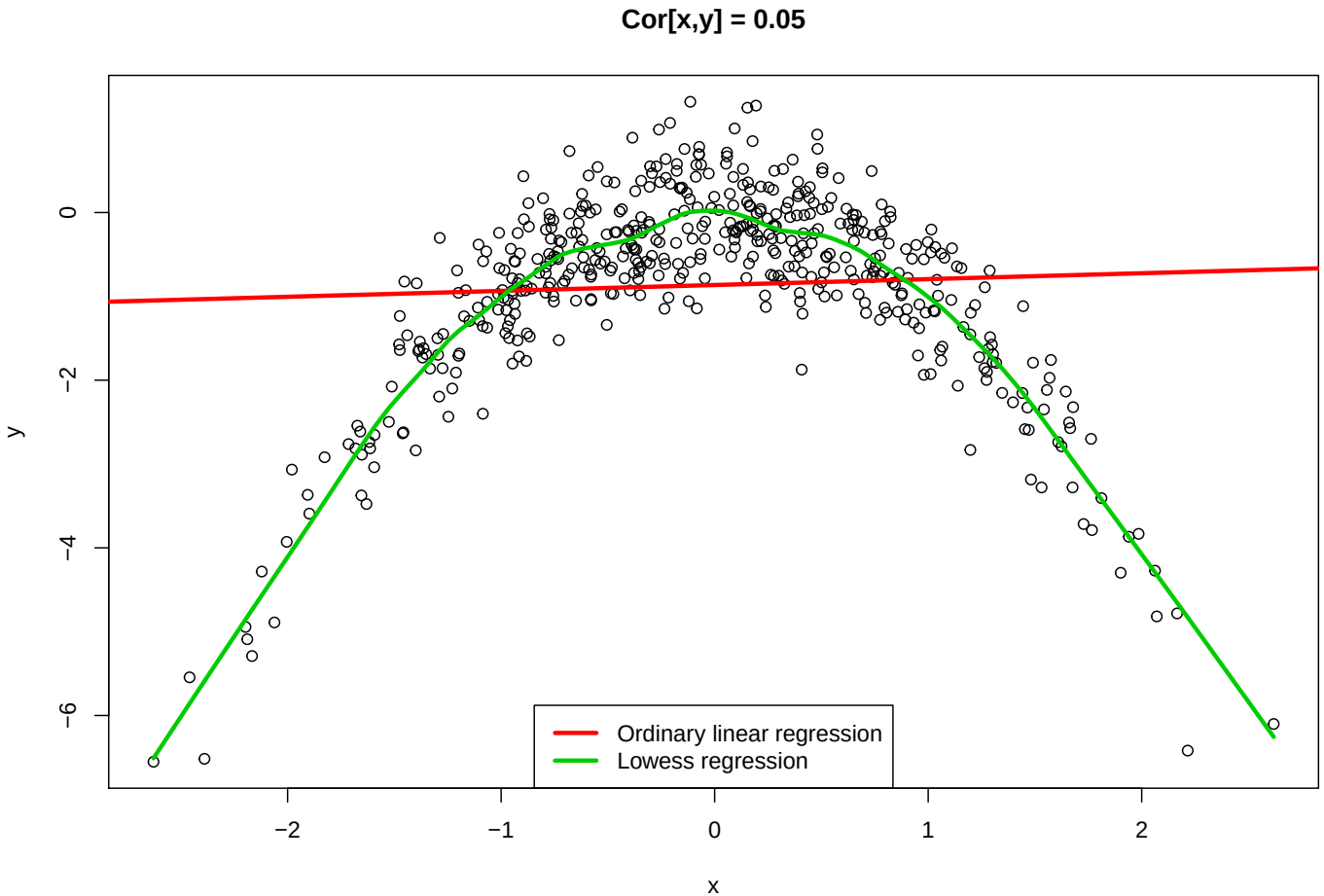








The importance of EDA



Exercise: Displaying high-dimensional data

What if we have more than two variables?

Given the `golub` dataset:

1. Find the top 5 most variable genes (hint: function `rowVars` in the package `matrixStats`).
2. Visualize their expression levels (hint: functions `pairs`, `heatmap.2`)

Solution: Displaying high-dimensional data

```
library(matrixStats)
vars <- rowVars(golub)
names(vars) <- rownames(golub)
vars <- sort(vars, decreasing = TRUE)
golub5 <- golub[names(vars)[1:5],]
```

```
heatmap(golub5)
NMF::aheatmap(golub5, color = clusterExperiment::seqPal5,
              annCol = data.frame(Class=pheno))
pairs(t(golub5), pch=20, col=colors[pheno])
boxplot(t(golub5), las=2, ylab="Expression levels")

cors <- cor(t(golub5))
heatmap(cors, col=clusterExperiment::seqPal3)
```

Dimensionality reduction

Dimensionality reduction

Data consist of *variables recorded on observational units*.

The data for J variables and n observations can be represented as an $n \times J$ *data matrix* $X = (X_{i,j} : i = 1, \ldots, n; j = 1, \ldots, J)$.

In genomic settings, the number of variables J is often in the tens of thousands and the sample size $n \ll J$.

Dimensionality reduction

Dimensionality reduction, i.e., representing the data using *fewer than J variables*, is useful for *summarizing* and *visualizing* data, in the context of exploratory data analysis (EDA, e.g., detecting main features), quality assessment/control (QA/QC, e.g., detecting artifacts, outliers), and reporting of results (e.g., clusters).

A variety of often related approaches can be used, but we focus on PCA.

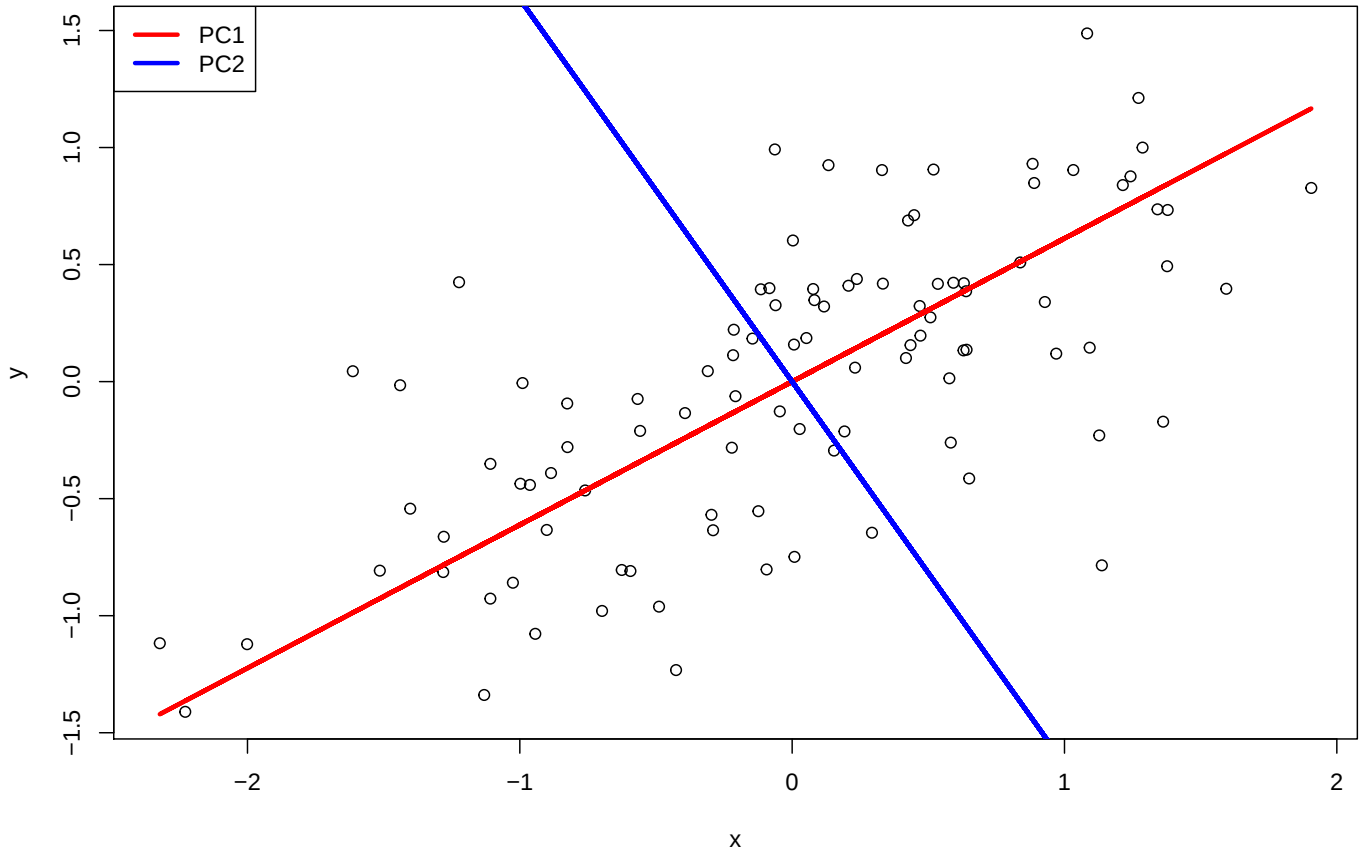
Principal component analysis (PCA) replaces the original variables by fewer *orthogonal linear combinations* of these variables, with successively *maximal variance* (Mardia, Kent, and Bibby 1979) (Chapter 8).

Principal Component Analysis

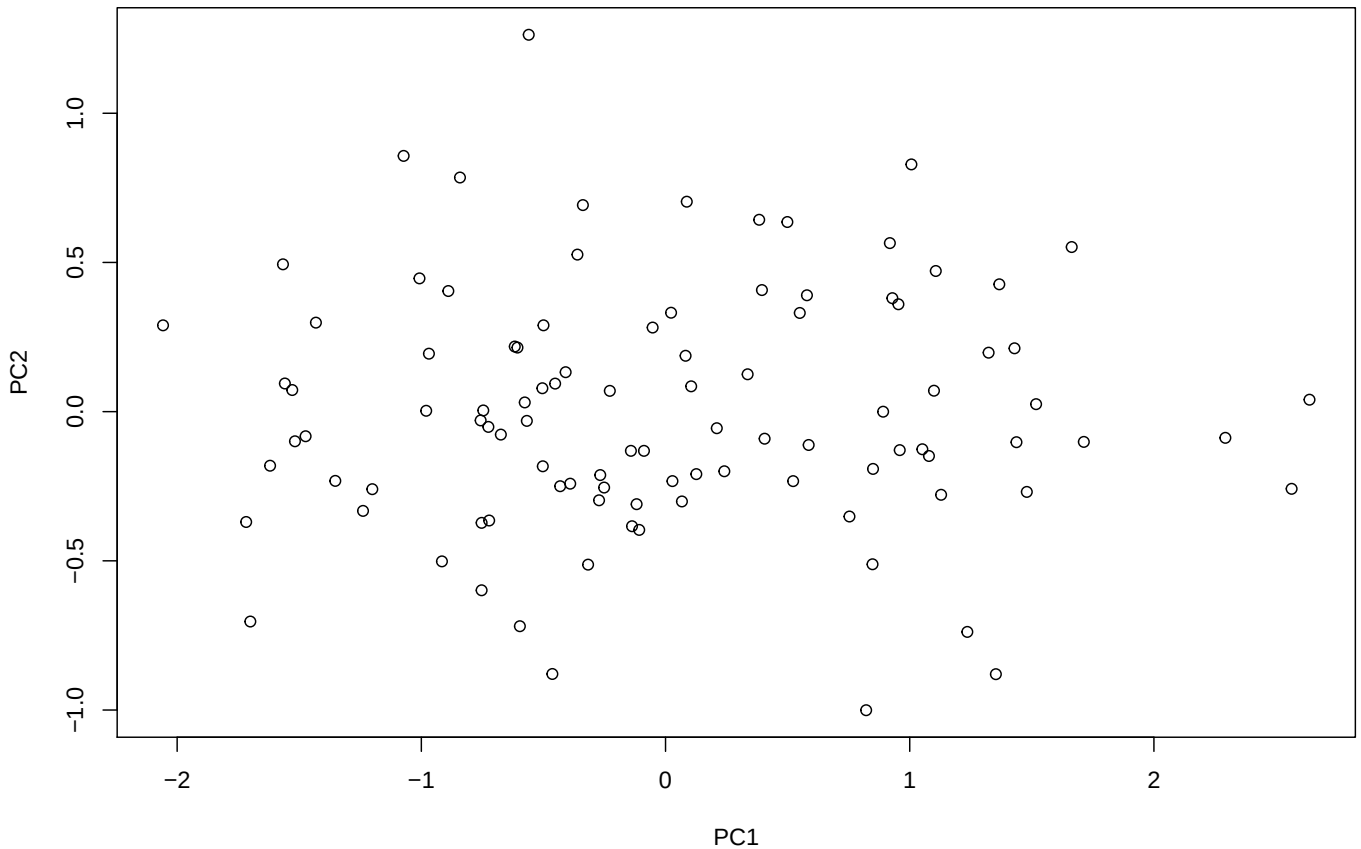
Principal component analysis (PCA) is a dimensionality reduction technique that provides a parsimonious summarization of the data by replacing the original variables by fewer *linear combinations* of these variables, that are *orthogonal* and have successively *maximal variance*.

Such linear combinations seek to "separate out" the observations, while loosing as little information as possible.

Principal Component Analysis



Principal Component Analysis



More formally

Consider a J-dimensional random vector $X = (X_j : j = 1, \ldots, J) \in \mathbb{R}^J$, with distribution P, mean vector $\mu = \mathrm{E}[X]$, and covariance matrix $\Sigma = \mathrm{Cov}[X] = \mathrm{E}[(X - \mu)(X - \mu)^\top]$.

The *principal component transformation* of the random vector X is defined as the standardized linear combination

$$X \to Y = \Gamma^\top (X - \mu), \tag{1}$$

where, from the Spectral Decomposition Theorem, $\Gamma^\top \Sigma \Gamma = \Lambda$, $\Lambda = \mathrm{Diag}(\lambda_j : j = 1, \ldots, J)$ is the diagonal matrix of eigenvalues of Σ, ordered as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_J \geq 0$, and $\Gamma = (\gamma_j : j = 1, \ldots, J)$ is an orthogonal matrix whose columns γ_j are the corresponding eigenvectors.

More formally

The *jth principal component* (jth PC or PC_j) of X is then

$$Y_j = \gamma_j^\top (X - \mu), \tag{2}$$

where the jth eigenvector γ_j is referred to as the *jth vector of principal component loadings*.

PCA properties

The principal components satisfy the following properties.

1. $$\mathrm{E}[Y] = 0_J,$$

   where 0_J is a J-dimensional column vector of zeroes.

2. $$\mathrm{Cov}[Y] = \Lambda = \mathrm{Diag}(\lambda_j : j = 1, \ldots, J)$$

   and, in particular,

   $$\mathrm{Var}[Y_1] \geq \mathrm{Var}[Y_2] \geq \ldots \geq \mathrm{Var}[Y_J].$$

3. $$\sum_{j=1}^{J} \mathrm{Var}[Y_j] = \mathrm{tr}\Sigma = \sum_{j=1}^{J} \lambda_j.$$

4. $$\prod_{j=1}^{J} \mathrm{Var}[Y_j] = |\Sigma| = \prod_{j=1}^{J} \lambda_j.$$

PCA properties

The principal components are *not scale-invariant*! Depending on the types of variables under consideration, scaling or use of the correlation matrix instead of the covariance matrix may be in order.

The *proportion of total variation explained* by the first K principal components is given by

$$p_K = \frac{\sum_{j=1}^{K} \lambda_j}{\sum_{j=1}^{J} \lambda_j}. \tag{3}$$

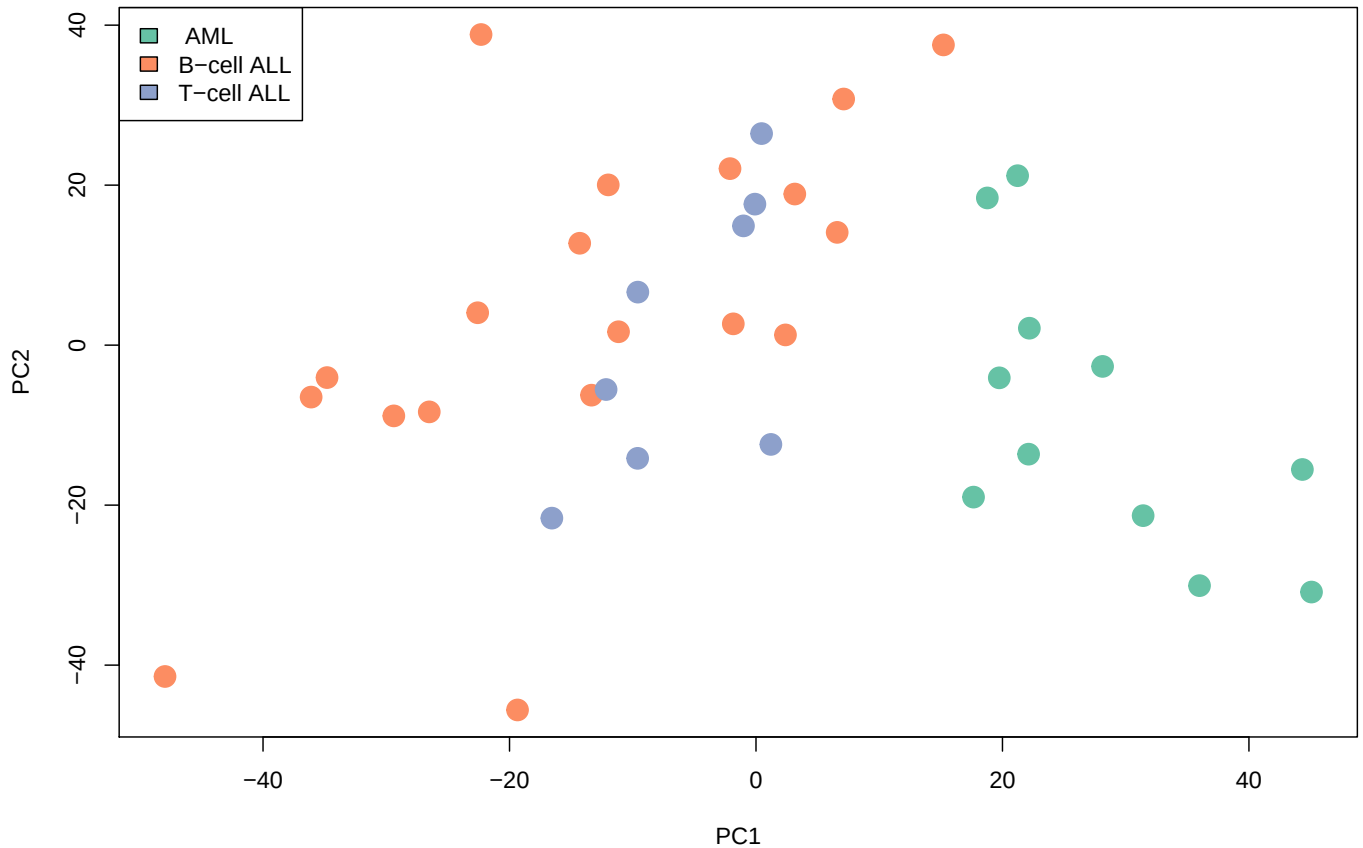
The *scree plot*, i.e., a plot of λ_j vs. j, can be used to determine how many components to use (stop at "elbow").

R software

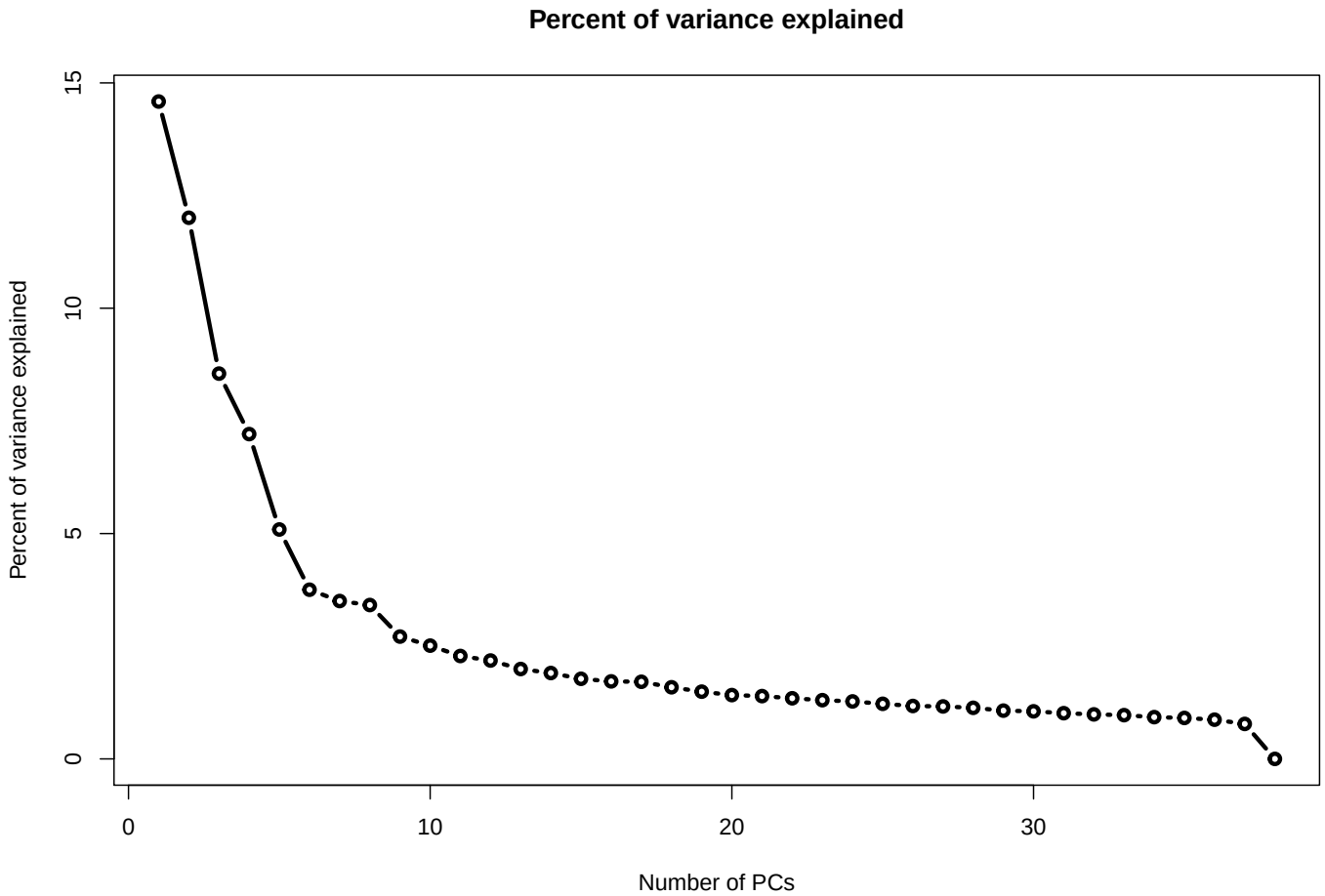
- `princomp` and `prcomp` can be used to compute the principal components.
- `svd` can be used to obtain the singular value decomposition.
- `eigen` to obtain the eigenvectors and eigenvalues.

Example

```
pca <- prcomp(t(golub))
plot(pca$x, pch=19, col=colors[pheno], cex=2)
legend("topleft", levels(pheno), fill=colors)
```



Example


Percent of variance explained
Percent of variance explained
0
5
10
15
0
10
20
30
Number of PCs

Classification

Class prediction

Consider a data structure $(X, Y) \sim P$, where $Y \in \{1, \ldots, K\}$ is a *scalar polychotomous outcome* (a.k.a., dependent variable, response) and $X = (X_j : j = 1, \ldots, J) \in \mathbb{R}^J$ is a J-dimensional vector of *covariates* (a.k.a., explanatory, feature, independent, or predictor variables).

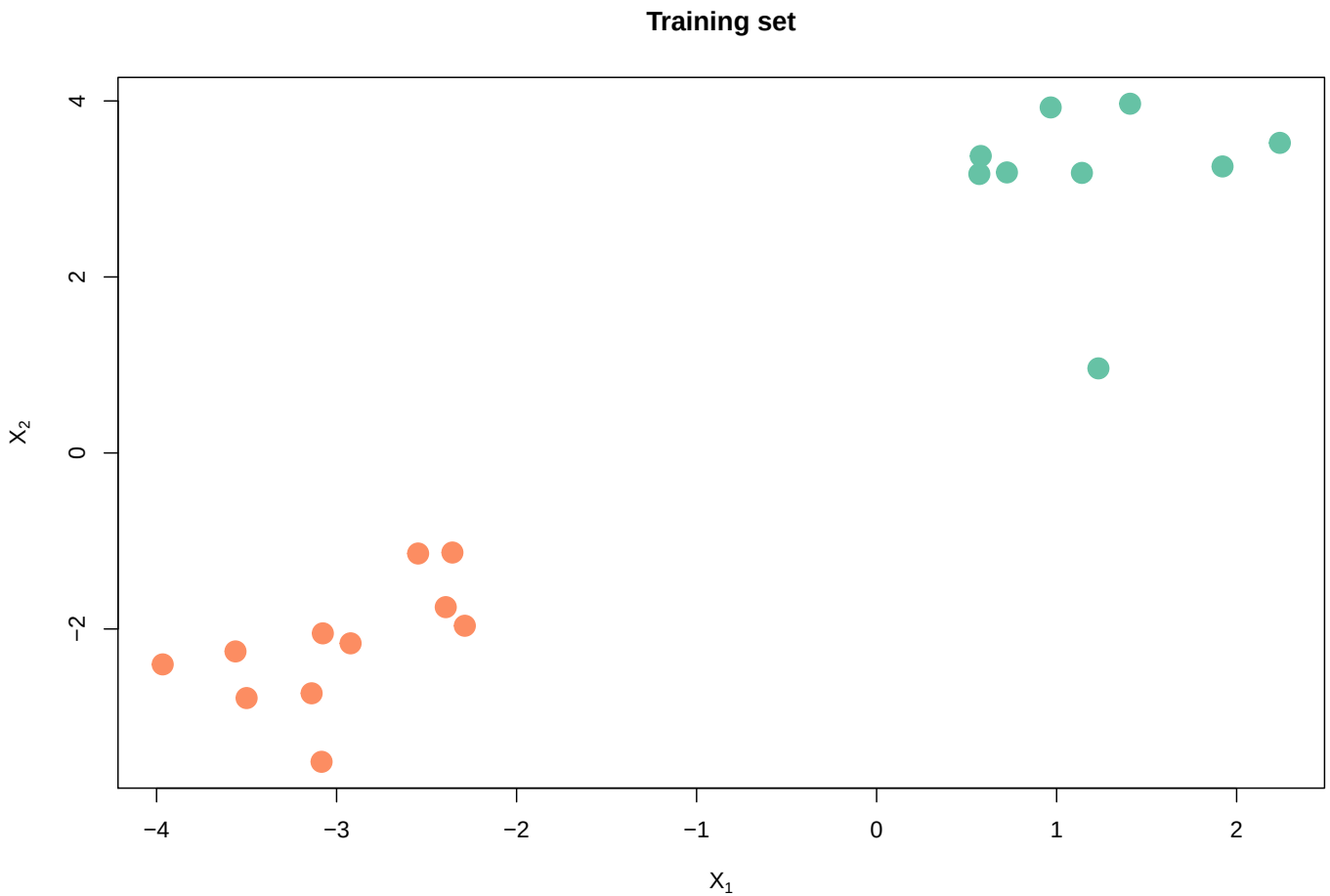
The task is to *classify* an observation, i.e., *predict class* Y, *given covariates* X.

E.g. Predict tumor class or response to treatment Y, given thousands of microarray or high-throughput sequencing measures of gene expression X.

A *classification function* or, in short, a *classifier*, is a mapping, $\psi : \mathcal{X} \to \{1, \ldots, K\}$, from a J-dimensional covariate space $\mathcal{X} \subseteq \mathbb{R}^J$ into the integers $\{1, \ldots, K\}$.

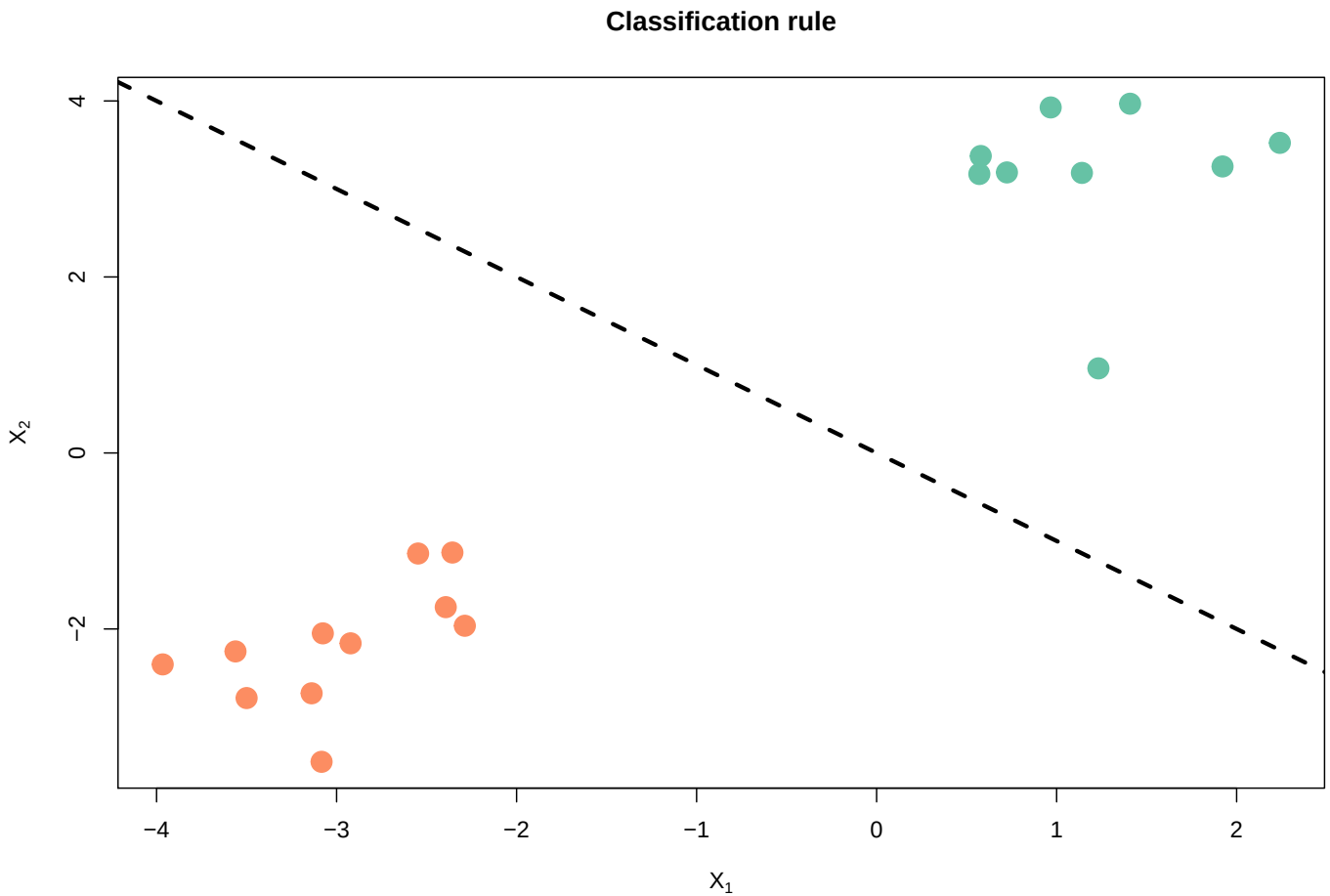
Toy example

Training set



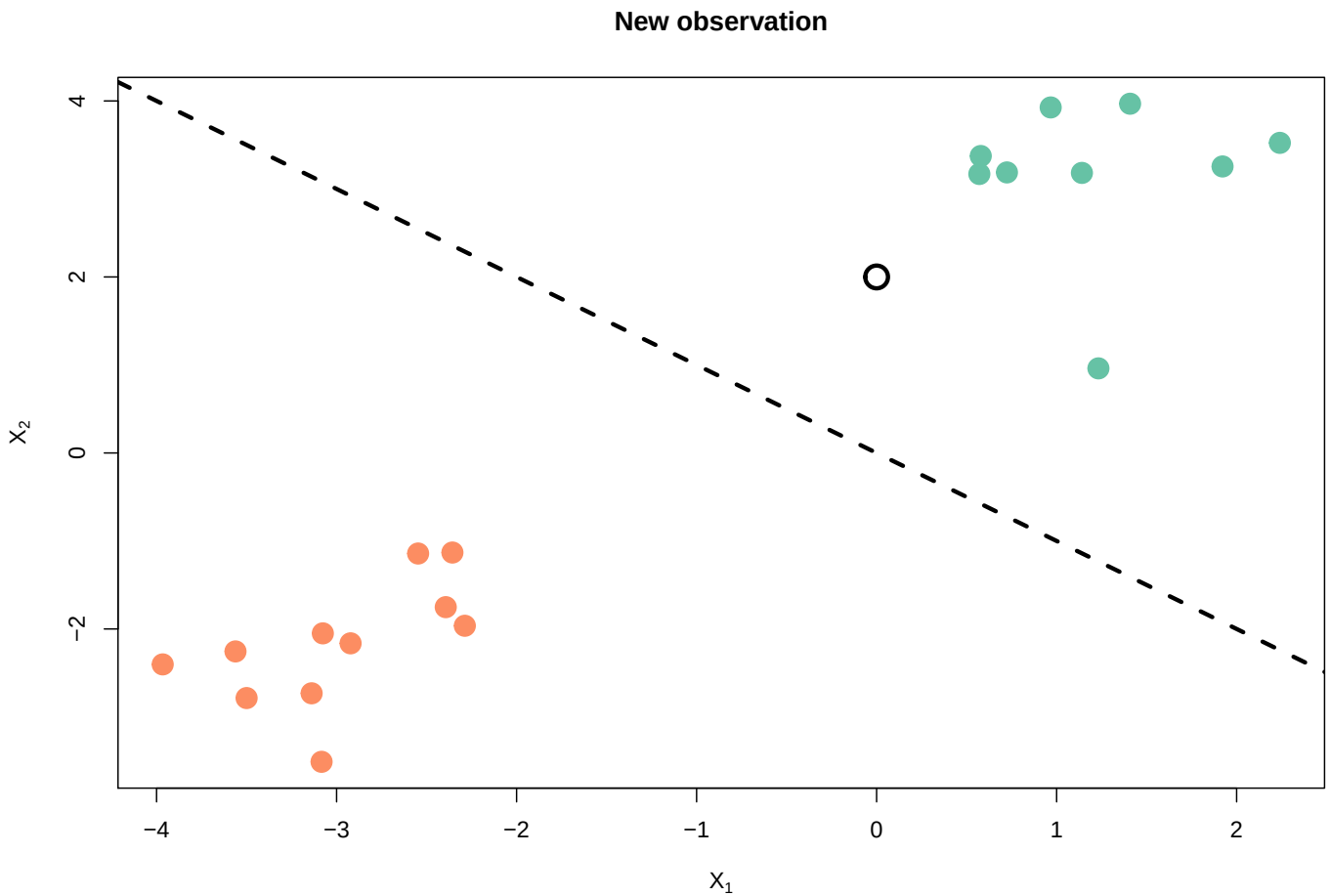
Toy example

Classification rule



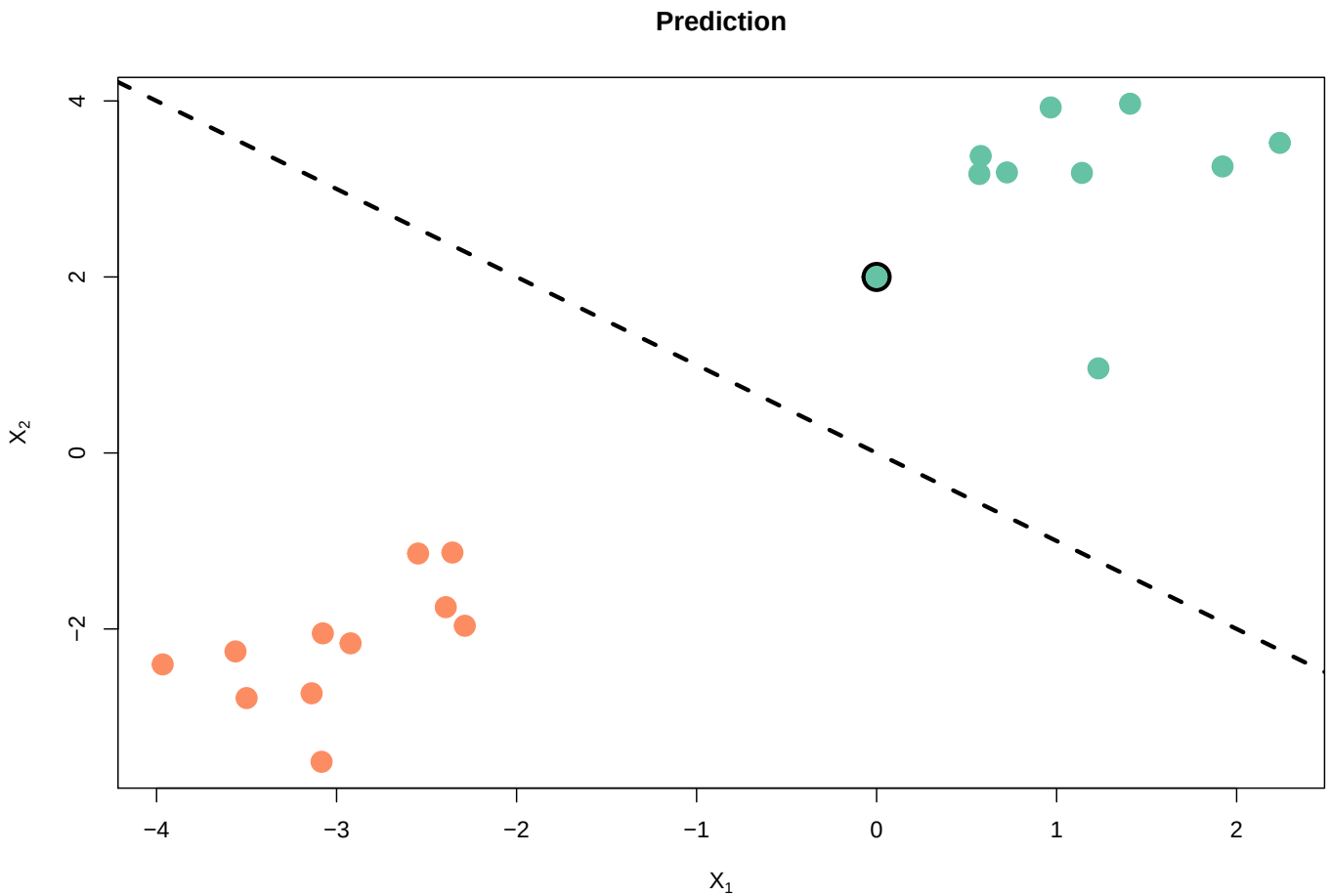
Toy example

New observation



Toy example

Prediction



Class prediction

A classifier generates a *partition of the covariate space* $\mathcal{X}$ into K disjoint and exhaustive subsets, $\mathcal{C}_1, \ldots, \mathcal{C}_K$, such that for an observation with covariates $X \in \mathcal{C}_k$ the predicted class is k. That is,

$$\psi(X) = \sum_{k=1}^{K} k \; \mathsf{I}(X \in \mathcal{C}_k). \tag{4}$$

As regression, classification can be handled within the framework of *loss-based inference*. That is, classifiers (parameters and estimators thereof) can be defined in terms of a *loss function* $L((X, Y), \psi)$ and its associated *risk function*

$$\Theta_L(\psi, P) \equiv \int L((x, y), \psi) dP(x, y). \tag{5}$$

Loss function

A widely-used loss function is the *indicator*,

$$\begin{aligned} L((X,Y),\psi) &= \mathsf{I}(Y \neq \psi(X)) \\ &= \begin{cases} 1, & \text{if } Y \neq \psi(X) \text{ [incorrect classification]} \\ 0, & \text{if } Y = \psi(X) \text{ [correct classification]} \end{cases}. \end{aligned} \tag{6}$$

The corresponding loss matrix $\mathbf{L}$ has diagonal elements equal to zero and off-diagonal elements equal to one. The *risk* function is the *misclassification probability*

$$\Theta_L(\psi, P) = \mathrm{E}[\mathsf{I}(Y \neq \psi(X))] = \Pr(Y \neq \psi(X)). \tag{7}$$

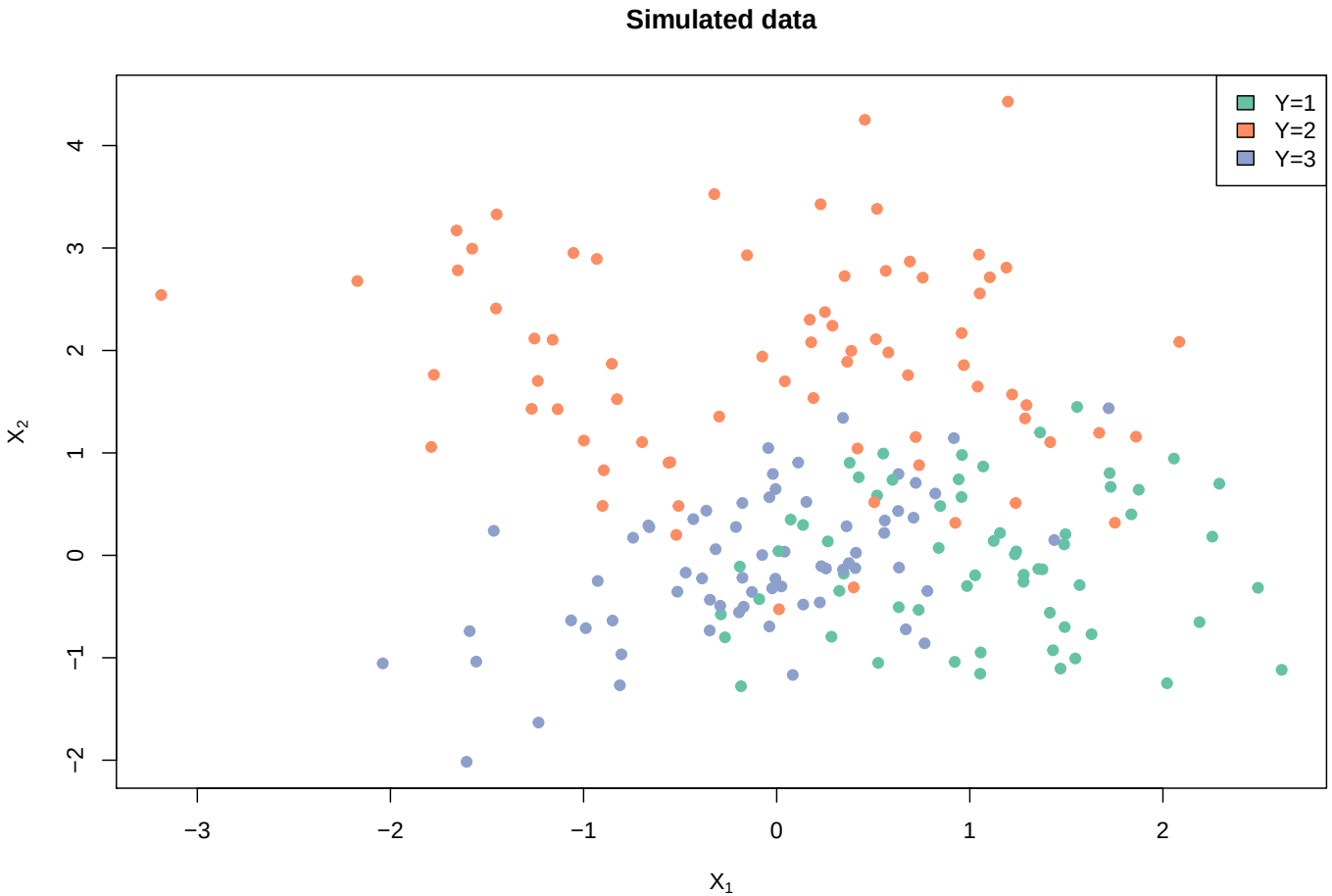
Note, that L does not need to be symmetric, e.g., it can be more harmful to have false negatives than false positives (or vice versa).

Example of classifier partitions

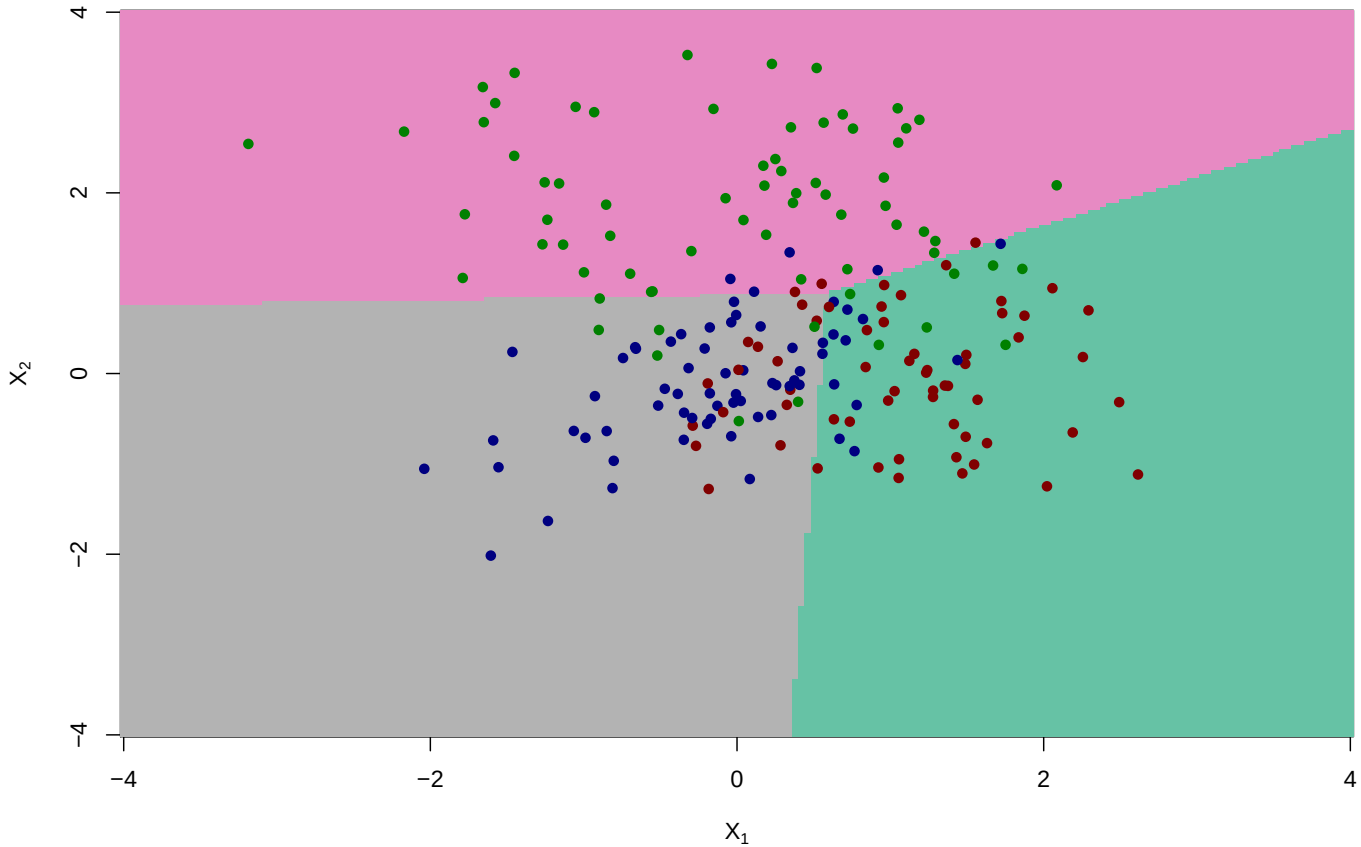
Simulate a learning set $\mathcal{L}_n = \{(X_i, Y_i) : i = 1, \ldots, n\}$ of $n = 200$ independent and identically distributed (IID) (X, Y) pairs, such that $Y \sim \mathcal{U}(\{1, \ldots, K\})$ and $X | Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$, $k = 1, \ldots, K$, with $K = 3$ classes and $J = 2$ covariates.

The following figures compare the *partitions* produced by different classifiers based on the learning set $\mathcal{L}_n$: *linear discriminant analysis* (LDA), *quadratic discriminant analysis* (QDA), *naive Bayes*, *k-nearest-neighbor* (k-NN), with $k = 1, 3, 5$.

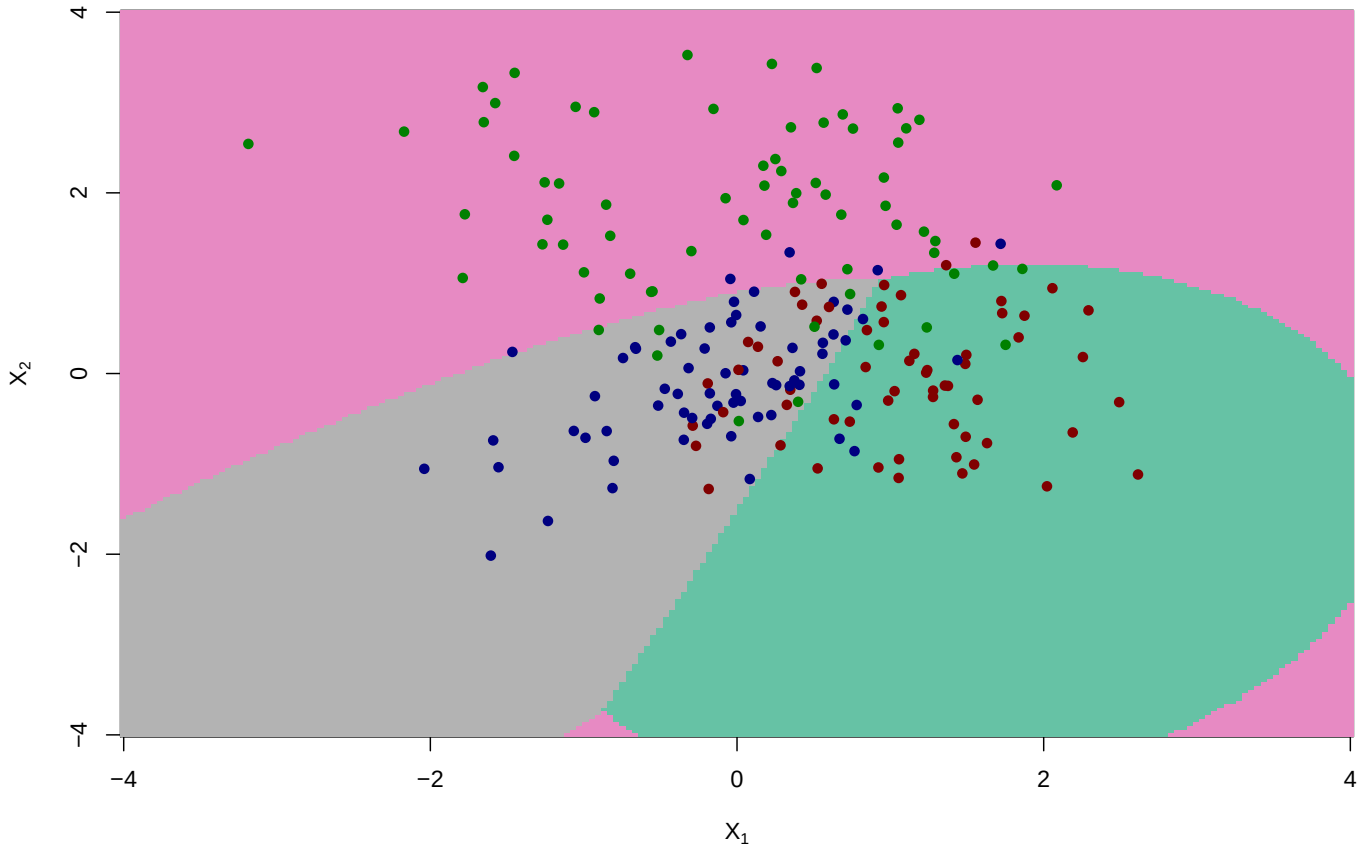
Example of classifier partitions



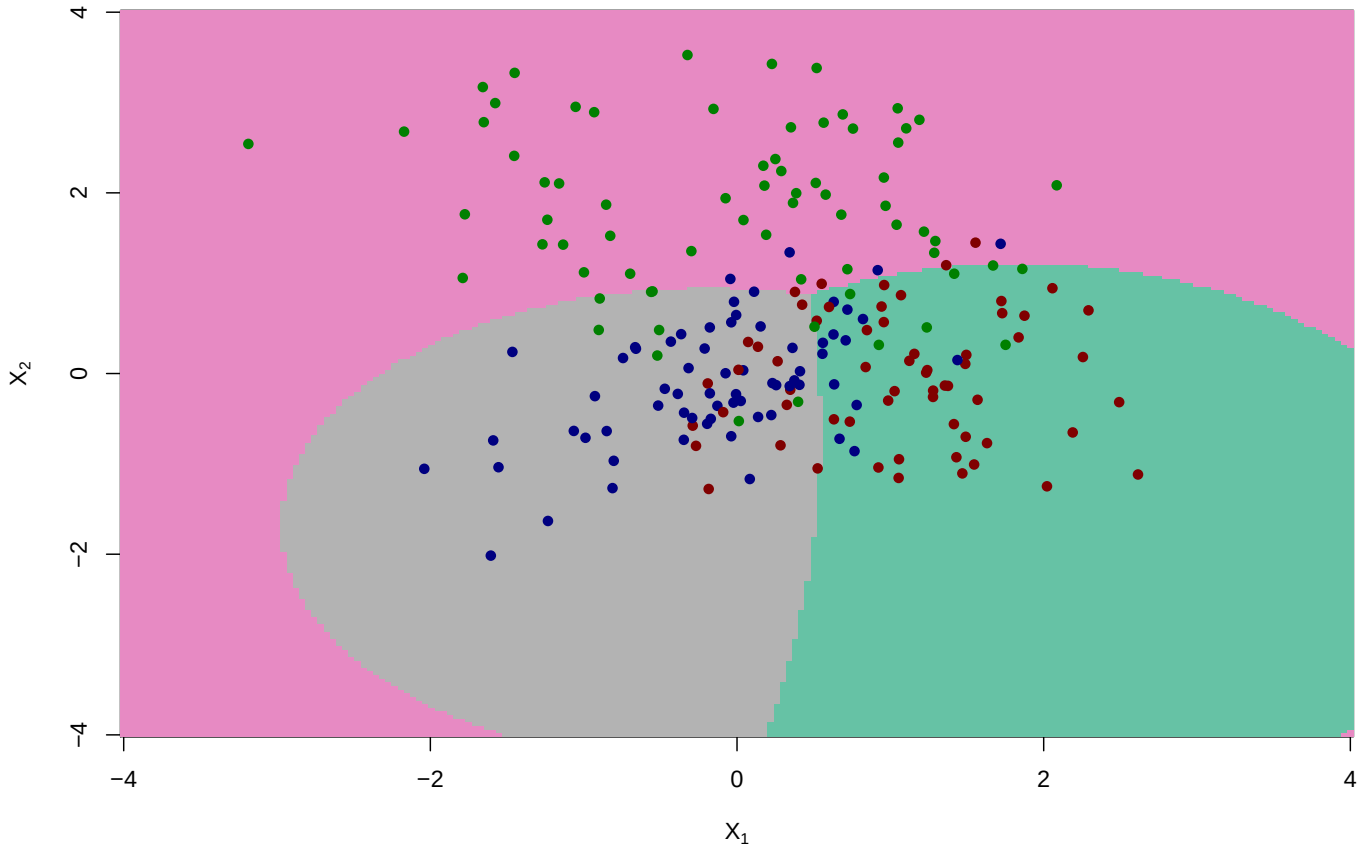
Example of classifier partitions: LDA



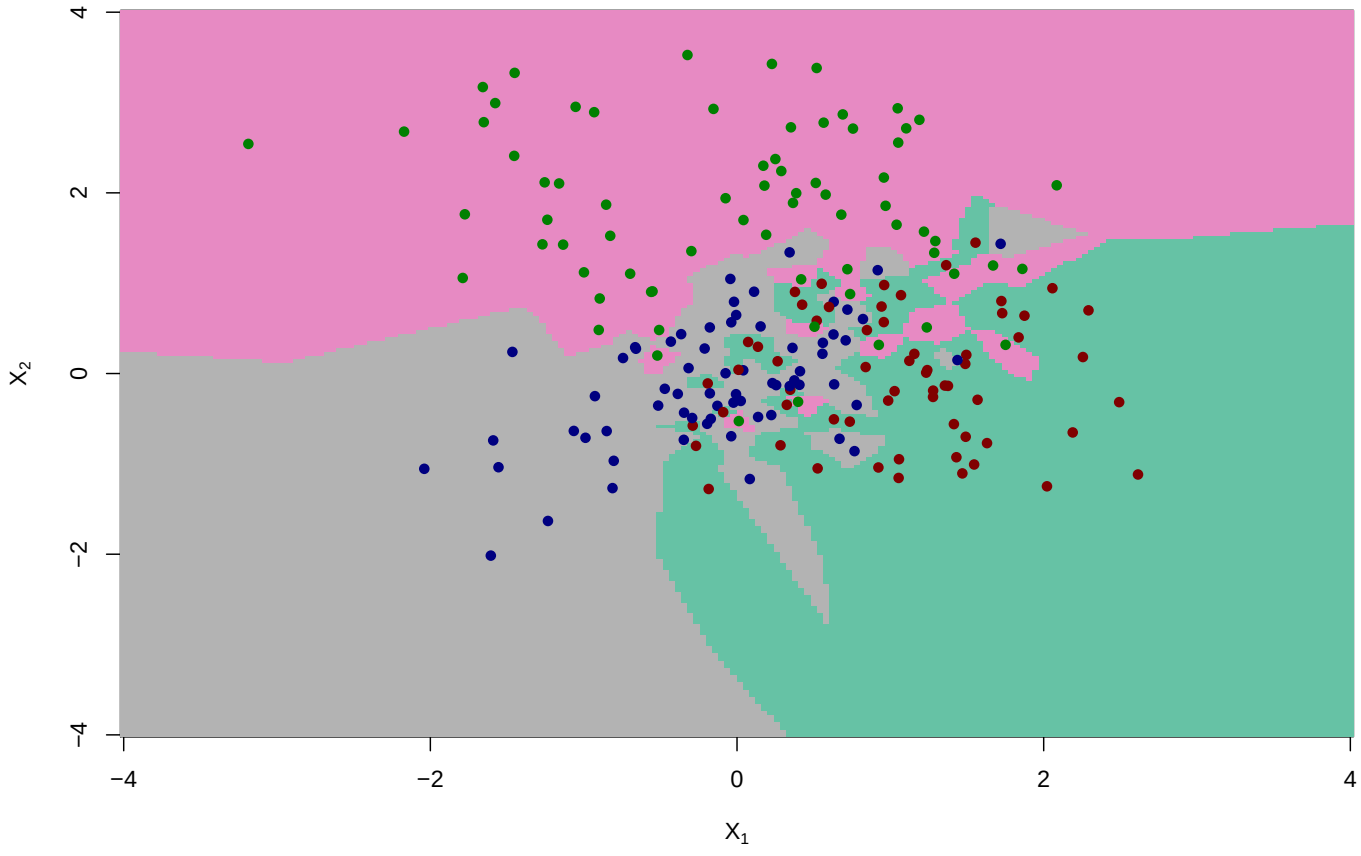
Example of classifier partitions: QDA



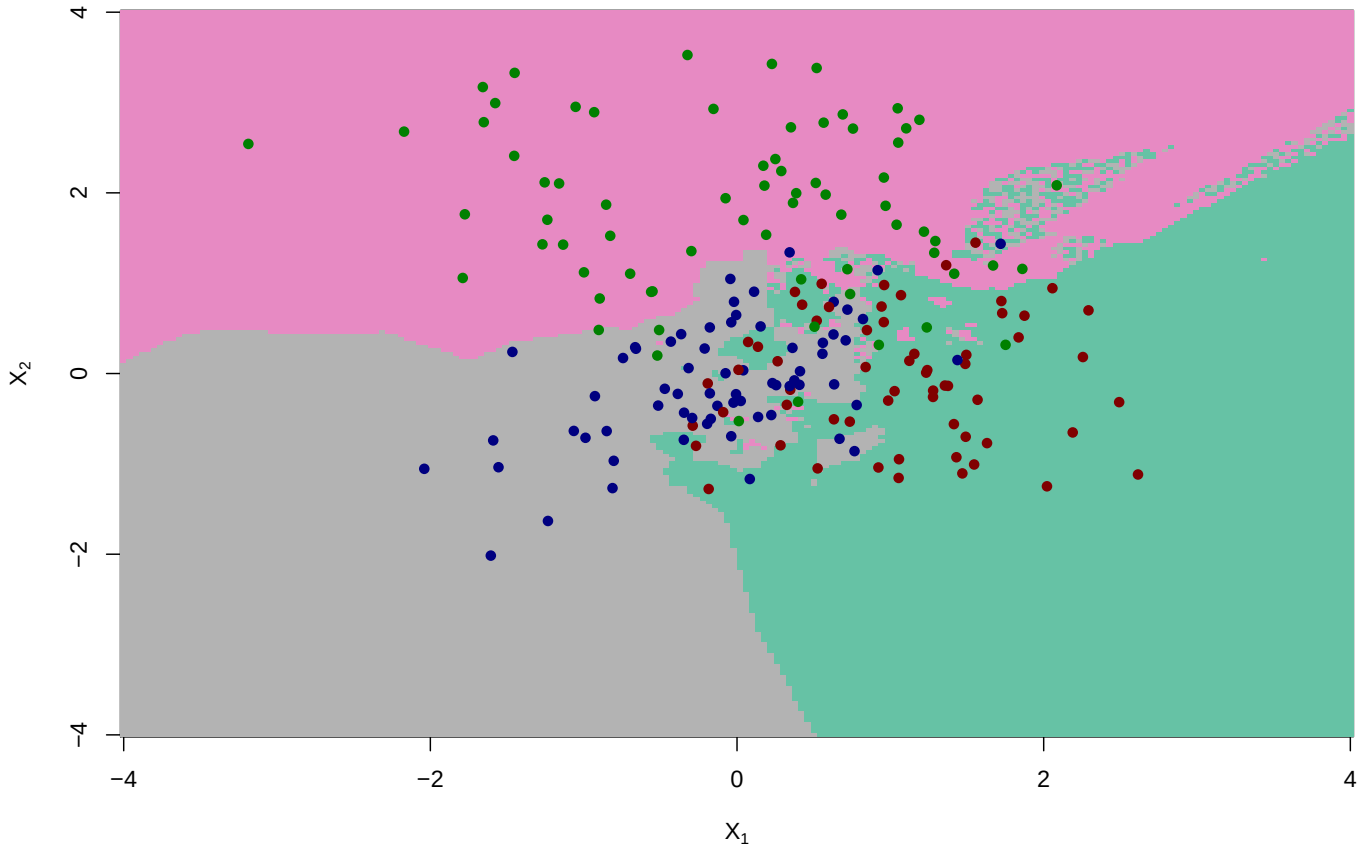
Example of classifier partitions: NB



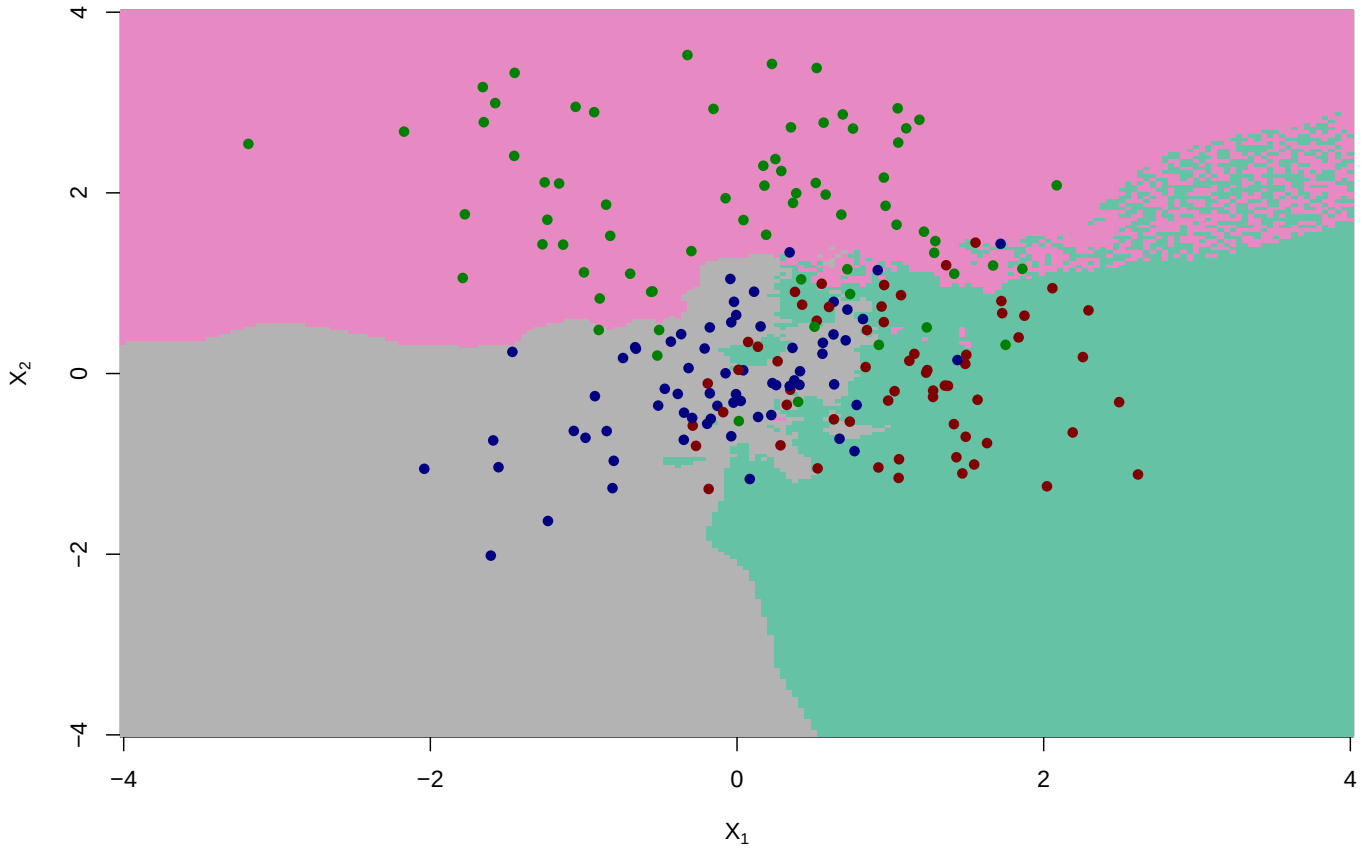
Example of classifier partitions: k-NN (k=1)



Example of classifier partitions: k-NN (k=3)



Example of classifier partitions: k-NN (k=5)



Class prediction vs. cluster analysis

- *Classification*. *Assign observational units to classes* on the basis of variables characterizing/describing these observations.
- *Cluster analysis.* The *classes are undefined* a priori and need to be "discovered" from the data.
    - a.k.a. Class discovery, unsupervised learning, unsupervised pattern recognition.
- *Class prediction.* The *classes are predefined* and the task is to understand the basis for the classification from a set of labeled observations, i.e., a learning set. This information is then used to *predict the class of future observations*.
    - a.k.a. Classification, discriminant analysis, supervised learning, supervised pattern recognition.

General issues in class prediction

- *Variable selection*. Filtering out uninteresting features.
- *Standardization of observations and/or variables* (cf. normalization).
- *Distance measure.*
- *Loss function.*
- *Class representation.*
- *Imputation of missing data.*
- *Polychotomous classification.* For classifiers that can only handle binary outcomes.
- *Confidence in prediction.*

Performance assessment

The *true risk* is typically unknown, as it depends on the unknown population distribution.

The *empirical risk*, i.e., the risk estimated from the learning set is *biased* and leads to *overfitting*.

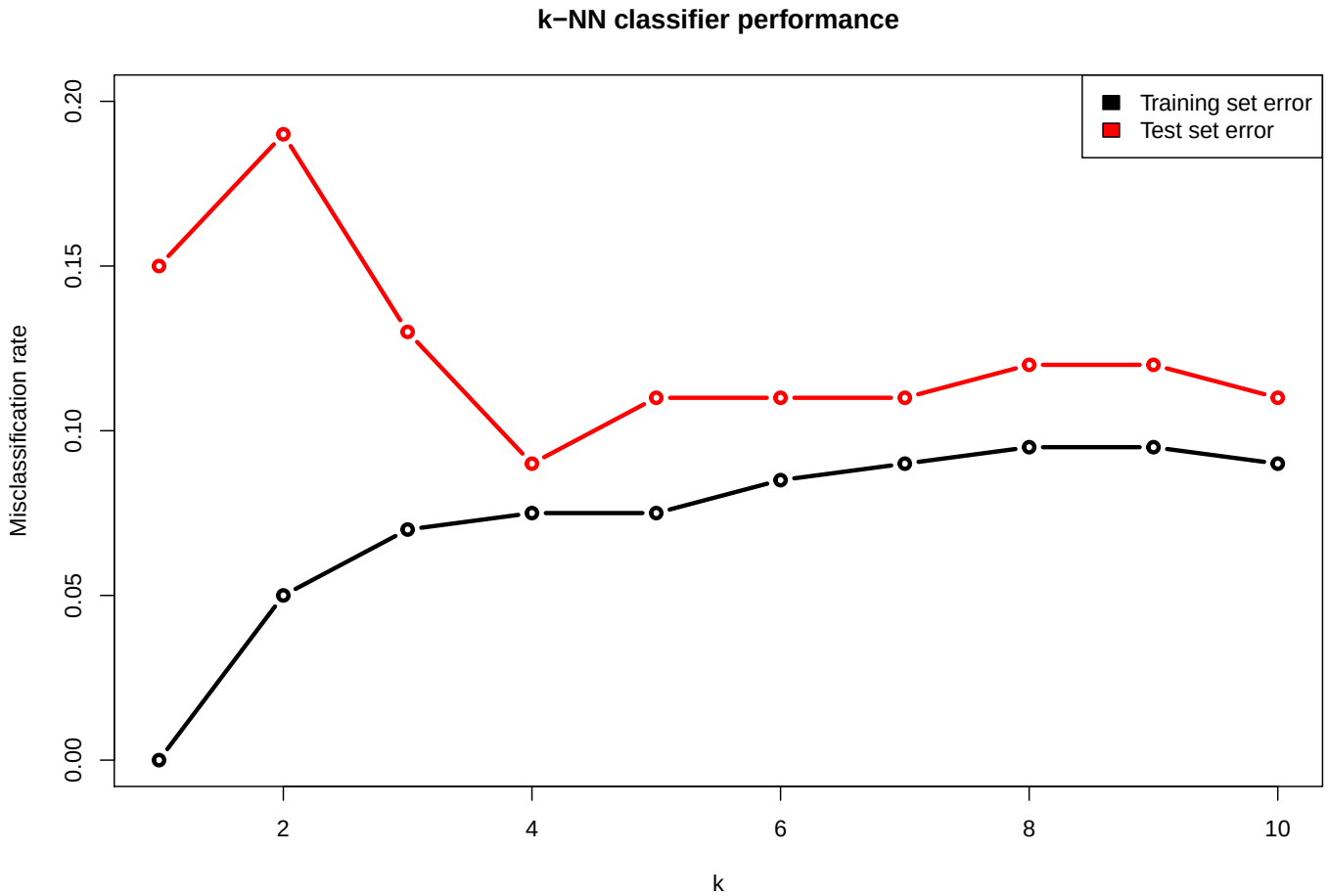
The classifier should be *trained* on the training set and tested on a different *test set* of observations.

In the absence of a genuine test set, one can use *cross-validation*.

Performance assessment

In order to provide accurate estimators of risk, it is essential to account for *all aspects* of the classifier learning process, e.g., variable selection, selection of number of neighbors k and distance function for k-nearest-neighbor (k-NN) classifiers.

Training vs test error



Bayes rule and classification

We are interested in the posterior probability

$$P(Y = k|X) = \frac{P(Y = k)\, P(X|Y = k)}{\sum_{i=1}^{K} P(Y = i)\, P(X|Y = i)}$$

The *classification rule* is to assign the observation to the population with the greatest posterior probability.

If we assume that the joint distribution of X is a multivariate normal, this gives rise to *Linear Discriminant Analysis* (LDA) and *Quadratic Discriminant Analysis* (QDA).

LDA assumes homogeneous variance across the populations, while QDA is the more general case of heterogeneous variances.

Quadratic Discriminant Analysis

Let us assume

$$P(X|\,Y=k) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp -\frac{1}{2}(X-\mu_k)^T\Sigma_k^{-1}(X-\mu_k).$$

Hence we need to minimize

$$\psi(X) = \arg\min_k \left\{(X-\mu_k)^\top\Sigma_k^{-1}(X-\mu_k) + \log|\Sigma_k| - 2\log p(k)\right\}.$$

The main quantity in the discriminant rule is $(X-\mu_k)^\top\Sigma_k^{-1}(X-\mu_k)$, the *squared Mahalanobis distance* from the covariate vector X to the class k mean vector μ_k.

Thus, intuitively, the predicted class for an observation with covariates X is the *class with closest mean vector* μ_k, for a suitably-defined distance function.

Linear Discriminant Analysis

If we assume that $\Sigma_k = \Sigma$ for all k, we have LDA.
The discriminant rule is based on the square of the Mahalanobis distance and is *linear in the covariates* X,

$$\begin{aligned} \psi(X) &= \arg\min_k \ (X-\mu_k)^\top \Sigma^{-1}(X-\mu_k) \\ &= \arg\min_k \ -2\mu_k^\top \Sigma^{-1} X + \mu_k^\top \Sigma^{-1}\mu_k. \end{aligned} \tag{8}$$

Special cases

- *Diagonal Linear Discriminant Analysis* (DLDA). If we assume a diagonal covariance matrix, $\Sigma_k = \Delta = \text{Diag}(\sigma_1^2, \ldots, \sigma_J^2)$, a.k.a. *naive Bayes rule* for Gaussian class conditional densities.
- *Nearest Centroids*. If we assume that $\Sigma_k = I_J$, the $J \times J$ identity matrix. Observations are classified on the basis of their *Euclidean distances from class means* μ_k.

Advantages of LDA and QDA

LDA and QDA are

- *Simple and intuitive*: The predicted class of a new observation is the class with the closest mean (using the Mahalanobis metric).
- *Optimal when the model is true*: LDA is based on the estimated Bayes rule for Gaussian class conditional densities and uniform priors.
- *Easy to implement*: LDA has linear boundaries.
- *Good performance in practice*: In spite of a possibly high bias, the low variance of the LDA estimators of class posterior probabilities often results in low classification error.

Limitations

- Linear or even quadratic discriminant boundaries may not be flexible enough.
- In the case of a high-dimensional covariate space, performance may degrade rapidly due to over-parameterization and high variance of estimators.

Extensions

- *Flexible discriminant analysis* (FDA). LDA on transformed outcomes and/or covariates.
- *Penalized discriminant analysis* (PDA). A penalized Mahalanobis distance is used to enforce smoothness.
- *Mixture discriminant analysis* (MDA). Class conditional densities are modeled as mixtures of Gaussian densities.
- *Automatic variable selection*. Shrinkage methods can be applied to perform automatic variable selection.

See Ripley (1996) and Hastie, Tibshirani, and Friedman (2001).

R software

- `lda`, `predict.lda` (`MASS`): Linear discriminant analysis.
- `qda`, `predict.qda` (`MASS`): Quadratic discriminant analysis.
- `naiveBayes` (`e1071`): Diagonal linear discriminant analysis, naive Bayes.

Many arguments and values; consult help files for details and examples.

Exercise: A classifier for Leukemia samples

```
library(limma)

# Train data
data(Golub_Train)
golub<-exprs(Golub_Train)
golub[golub<100]<-100
golub[golub>16000]<-16000
log2(golub) %>% normalizeQuantiles -> golub
paste(Golub_Train$T.B.cell, Golub_Train$ALL.AML) %>%
  sub("NA", "", .) %>% as.factor -> pheno

# Test data
data(Golub_Test)
golubt<-exprs(Golub_Test)
golubt[golubt<100]<-100
golubt[golubt>16000]<-16000
log2(golubt) %>% normalizeQuantiles -> golubt
paste(Golub_Test$T.B.cell, Golub_Test$ALL.AML) %>%
  sub("NA", "", .) %>% as.factor -> phenot
```

Exercise: A classifier for Leukemia samples

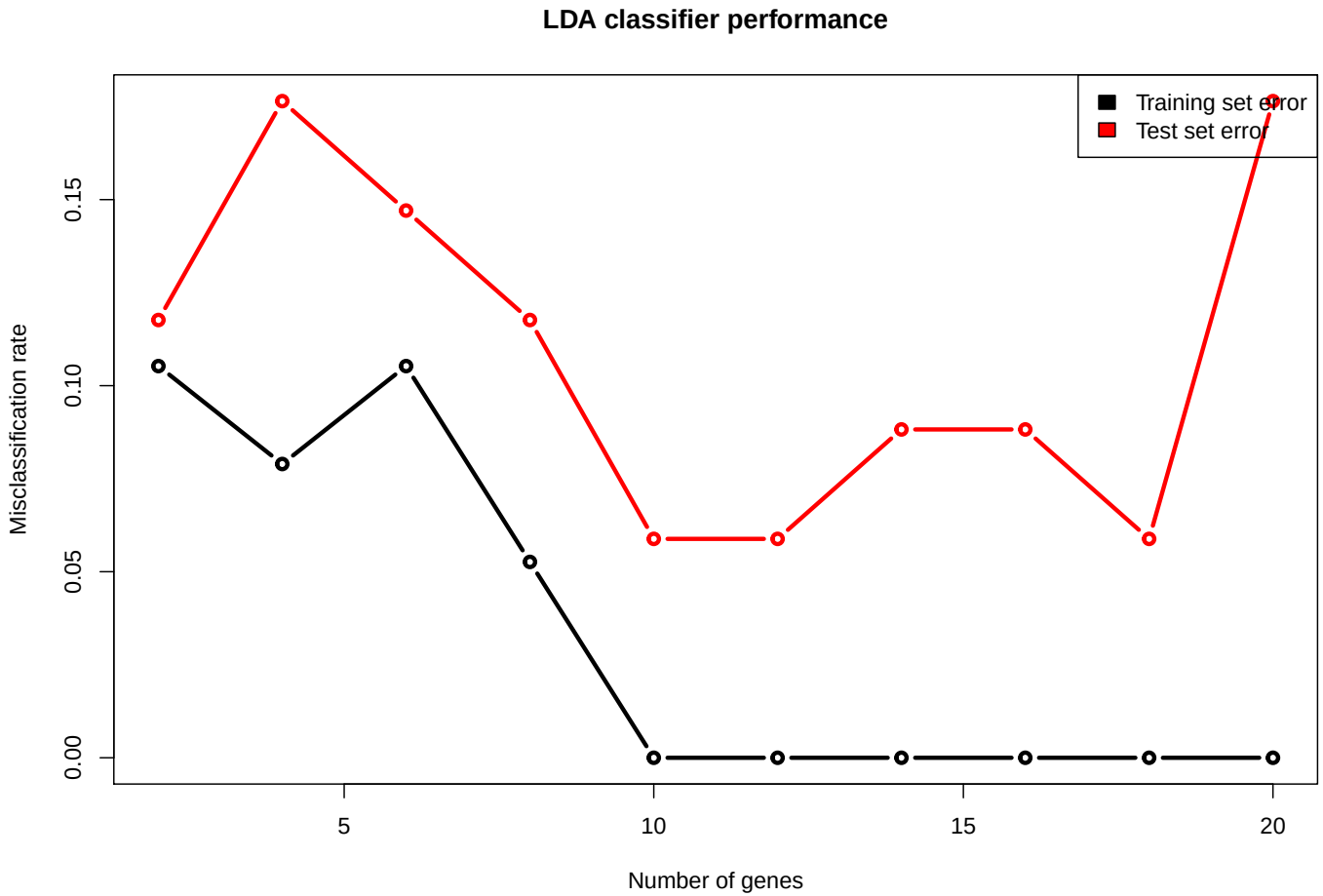
1. Find/Retrieve the most variable genes.
2. Use the top 2, 4, . . . , 20 genes to classify the leukemia samples with LDA.
3. Evaluate the performance both in the training and in the test set.
4. Plot the training and test set error as a function of the number of genes.

Solution: A classifier for Leukemia samples

```
vars <- rowVars(golub)
names(vars) <- rownames(golub)
vars <- sort(vars, decreasing = TRUE)

ngenes <- seq(2, 20, by=2)
er1 <- er2 <- numeric(length(ngenes))
for(i in seq_along(ngenes)) {
dat <- data.frame(pheno, t(golub[names(vars)[seq_len(ngenes[i])],]))
newdat <- data.frame(phenot, t(golubt[names(vars)[seq_len(ngenes[i])],]
fit <- lda(pheno~., data=dat)
pred_ls <- predict(fit, dat)
pred_ts <- predict(fit, newdat)

er1[i] <- sum(pred_ls$class != pheno)/length(pheno)
er2[i] <- sum(pred_ts$class != phenot)/length(phenot)
}
```

Solution: A classifier for Leukemia samples

```
plot(ngenes, er1, type='b', ylab="Misclassification rate",
     main="LDA classifier performance", ylim=c(0, max(er2)),
     xlab="Number of genes", lwd=3)
lines(ngenes, er2, type='b', col=2, lwd=3)
legend("topright", c("Training set error", "Test set error"), fill=1:2)
```



Other classification algorithms

One of the most active area of research in Statistics and Machine Learning.
Some of the most used classifiers are:

- Nearest-neighbor classifiers (Fix and Hodges 1951)
- Support Vector Machines (SVM) (Vapnik 1998)
- Classification And Regression Trees (CART) (Breiman et al. 1984)
- Neural Networks (cf. Deep Learning) (Ripley 1996)

Wikipedia has 79 pages in the category "Classification algorithms"

Ensemble methods

Substantial gains in accuracy can be obtained from *ensemble methods*, i.e., by *aggregating predictors*.

For polychotomous outcomes, multiple predictors are aggregated by voting, for continuous outcomes, by averaging (Dudoit and Fridlyand 2003).

One can consider the *same predictor* built *on perturbed versions of the learning set*, e.g., bootstrap samples, or a *linear combination of/vote from different predictors*.

Ensemble methods

Examples include

- *Bagging*: Aggregate same predictor built on multiple bootstrap samples of the learning set (Breiman 1996).
- *Boosting*: Aggregate same predictor built from repeated adaptive resampling of the learning set, where sampling weights are increased for observations most often misclassified (J. Friedman, Hastie, and Tibshirani 2000).
- *Random forests*: Aggregate trees built on multiple bootstrap samples of the learning set, where covariates are randomly selected for consideration at each node (Breiman 2001).
- *Super Learner*: Linear combination of different predictors, where coefficients in the combination are selected by cross-validation (Laan, Polley, and Hubbard 2007).

Cross-validation

Cross-validation (CV)

To assess the performance, one cannot use the same observations for both training and testing the classifier.

If one does not have a *genuine test or validation set*, she can *partition the available learning set* $\mathcal{X}_n = \{X_i : i = 1, \ldots, n\}$ into two sets: a *training set* and a *validation* set.

Observations in the training set $\mathcal{T}_n$ are used to compute, or *train*, an estimator(s).

Observations in the validation set $\mathcal{V}_n$ are used to assess the risk of, or *validate*, this estimator(s).

Note that CV can be applied in general estimation problems and is not limited to classification.

Leave-one-out cross-validation (LOOCV)

In *leave-one-out cross-validation* (LOOCV), each observation in the learning set is used in turn as the validation set and the remaining $(n-1)$ observations are used as the training set.

The corresponding distribution of the split vector V_n places probability $1/n$ on each of the n binary vectors $v_n^i = (v_n^i(i') = \mathsf{I}(i' = i) : i' = 1, \ldots, n),\ i = 1, \ldots, n.$

The proportion of observations in the validation sets is constant, $\Upsilon_n = 1/n.$

V-fold cross-validation

Given a user-supplied fixed number of folds V, V-fold cross-validation randomly partitions the learning set into V mutually exclusive and exhaustive sets of approximately equal size.

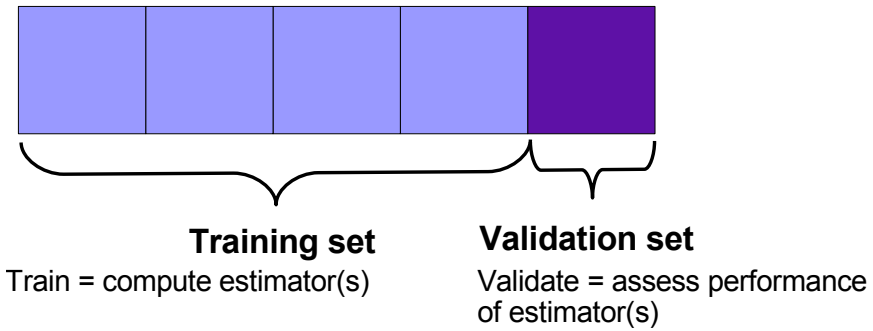
Each set is used in turn as the validation set.

Such a partition results in V binary vectors, $v_n^v = (v_n^v(i) : i = 1, \ldots, n)$, $v = 1, \ldots, V$, such that $\sum_i v_n^v(i) \approxeq n/V \ \forall \ v$ and $\sum_v v_n^v(i) = 1 \ \forall \ i$.

The corresponding distribution of the split vector V_n places probability $1/V$ on each of the V binary vectors v_n^v.

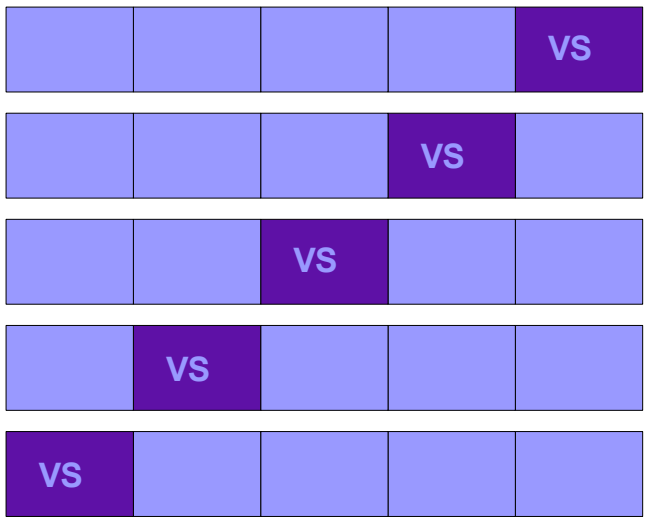
The proportion of observations in the V validation sets is $\Upsilon_n \approxeq 1/V$.

V-fold cross-validation



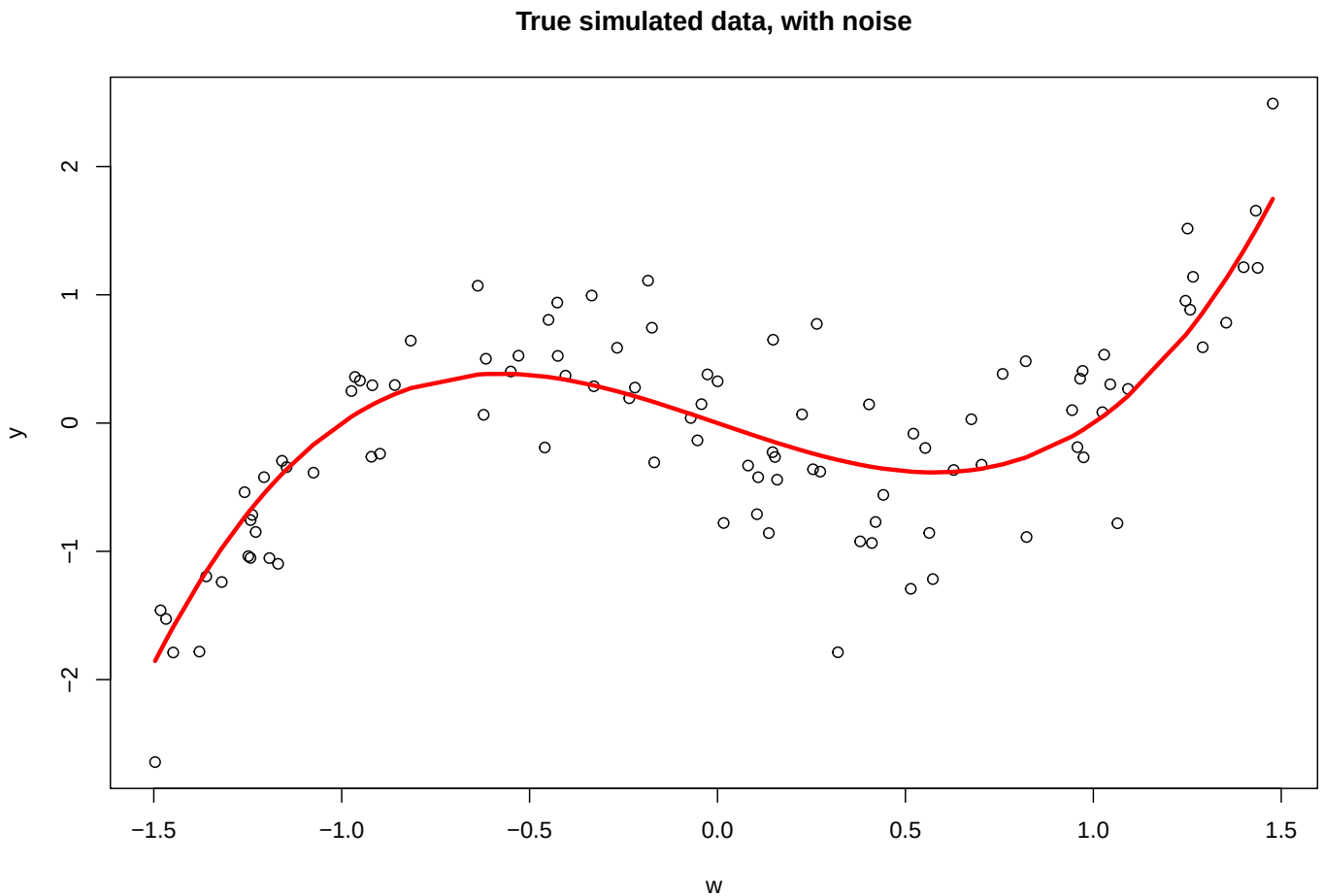
Training and validation sets in five-fold cross-validation.

V-fold cross-validation



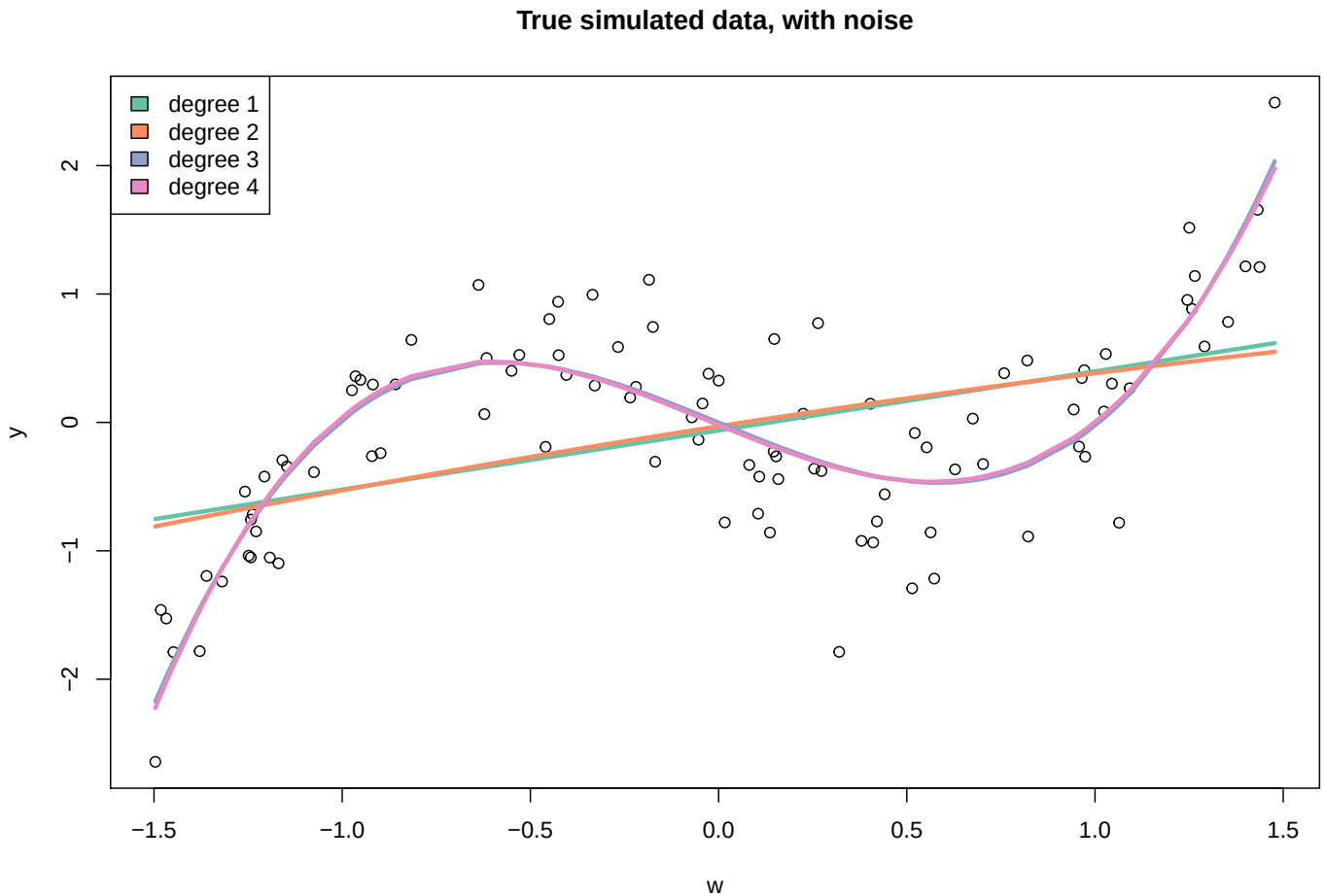
Training and validation sets in five-fold cross-validation.

Example: Polynomial regression



Example: Polynomial regression

How to select the degree of the regression function?



Example: Polynomial regression

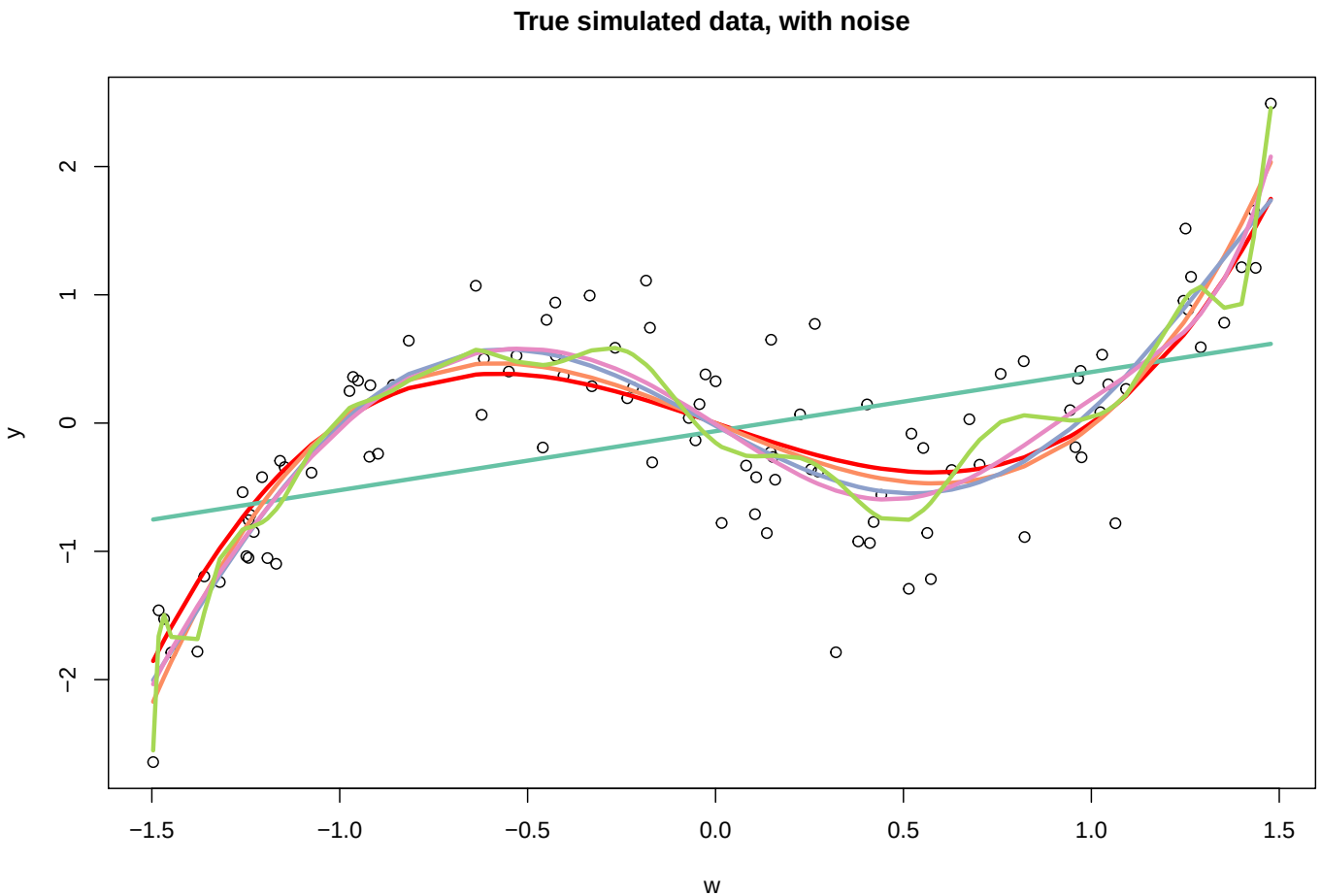
We want to minimize the *mean squared error* between the prediction and the true values.

```
risk <- function(pred, test) {
  mean((pred - test$Y)^2, na.rm=TRUE)
}
```

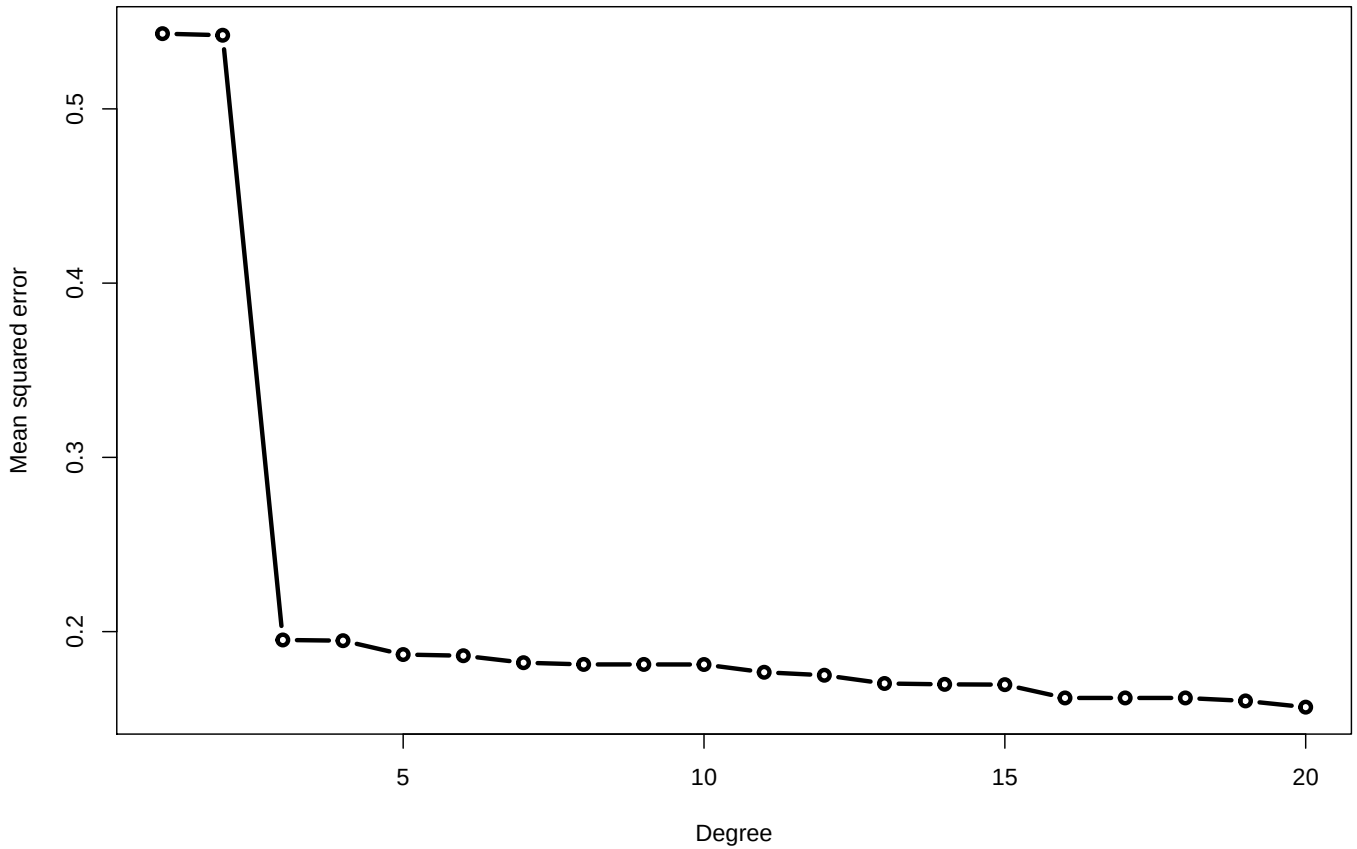
Example: Polynomial regression

```
predLM <- function(degree, train, test) {
  f <- c("Y ~ 1")
  for(dd in 1:degree) f <- paste(f,paste("I(W^",dd,")",sep=""),sep="+")
  fit <- lm(f, data=train)
  pred <- predict(fit, test)
  return(pred)
}

D <- 1:20
train <- data.frame(Y=y, W=w)

res <- lapply(D, function(d) predLM(d, train, train))
```

Example: Polynomial regression



Example: Polynomial regression



The problem is that if we train and test on the same data, we incur in *overfitting*.
Hence, we cannot use this as a criterion to choose the polynomial degree.

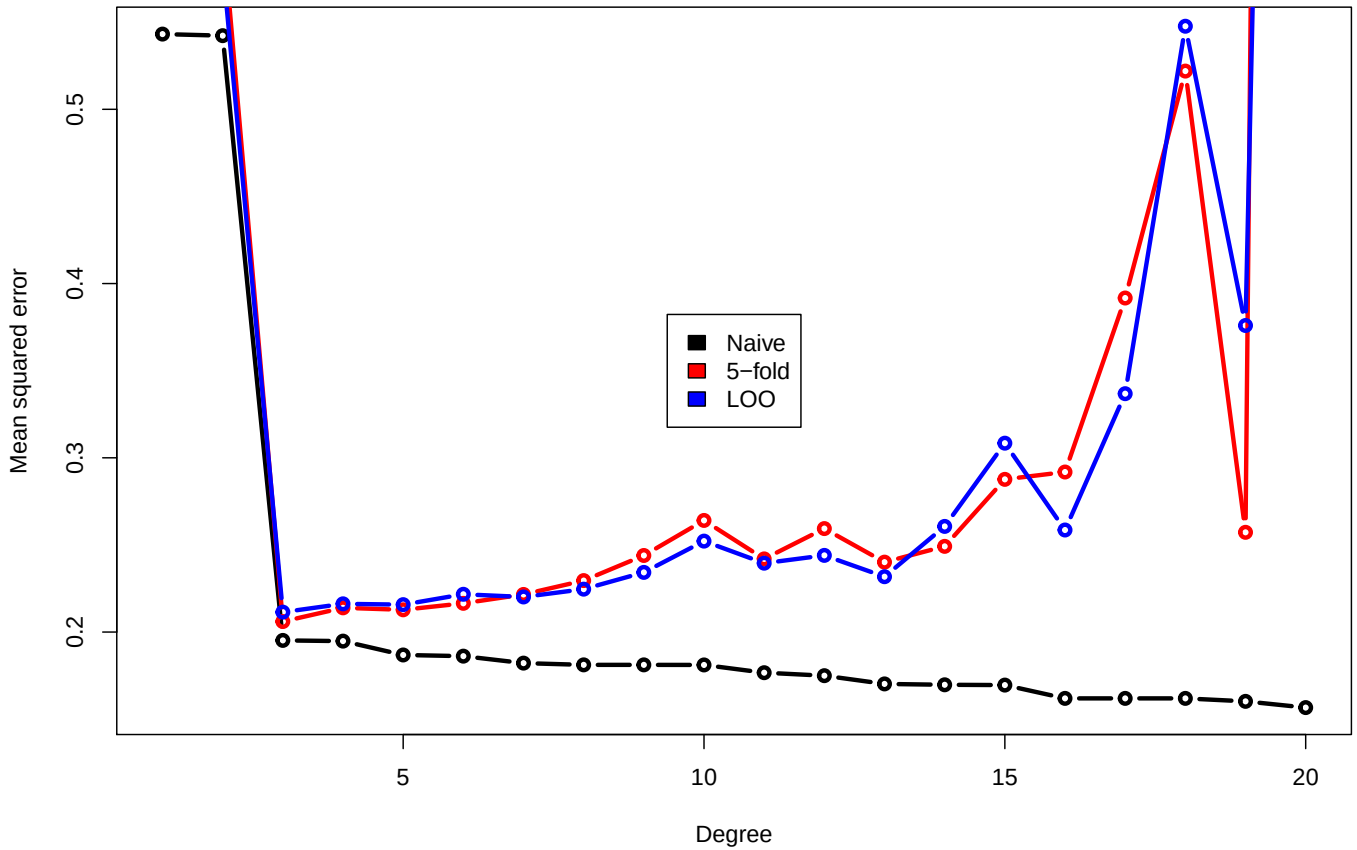
Example: 5-fold cross-validation

```
V <- 5
n <- floor(N/V)
fold <- sample(as.vector(mapply(rep, 1:V, n)))

cvres <- sapply(seq_len(V), function(v) {
  testSet <- data.frame(W=w[which(fold==v)], Y=y[which(fold==v)])
  trainSet <- data.frame(W=w[which(fold!=v)], Y=y[which(fold!=v)])

  lmRisk <- sapply(D, function(d) {
    pred <- predLM(d, trainSet, testSet)
    risk(pred, testSet)
  })

  return(lmRisk)
})

cvRiskLM <- rowMeans(cvres)
```

Example: LOO cross-validation

```
loores <- sapply(seq_len(N), function(v) {
  testSet <- data.frame(W=w[v], Y=y[v])
  trainSet <- data.frame(W=w[-v], Y=y[-v])

  lmRisk <- sapply(D, function(d) {
    pred <- predLM(d, trainSet, testSet)
    risk(pred, testSet)
  })

  return(lmRisk)
})

looRiskLM <- rowMeans(loores)
```

Example: Polynomial regression



Example: Polynomial regression

```
which.min(cvRiskLM)
```

```
## [1] 3
```

```
which.min(looRiskLM)
```

```
## [1] 3
```

Exercise: compute the CV error for the Leukemia dataset

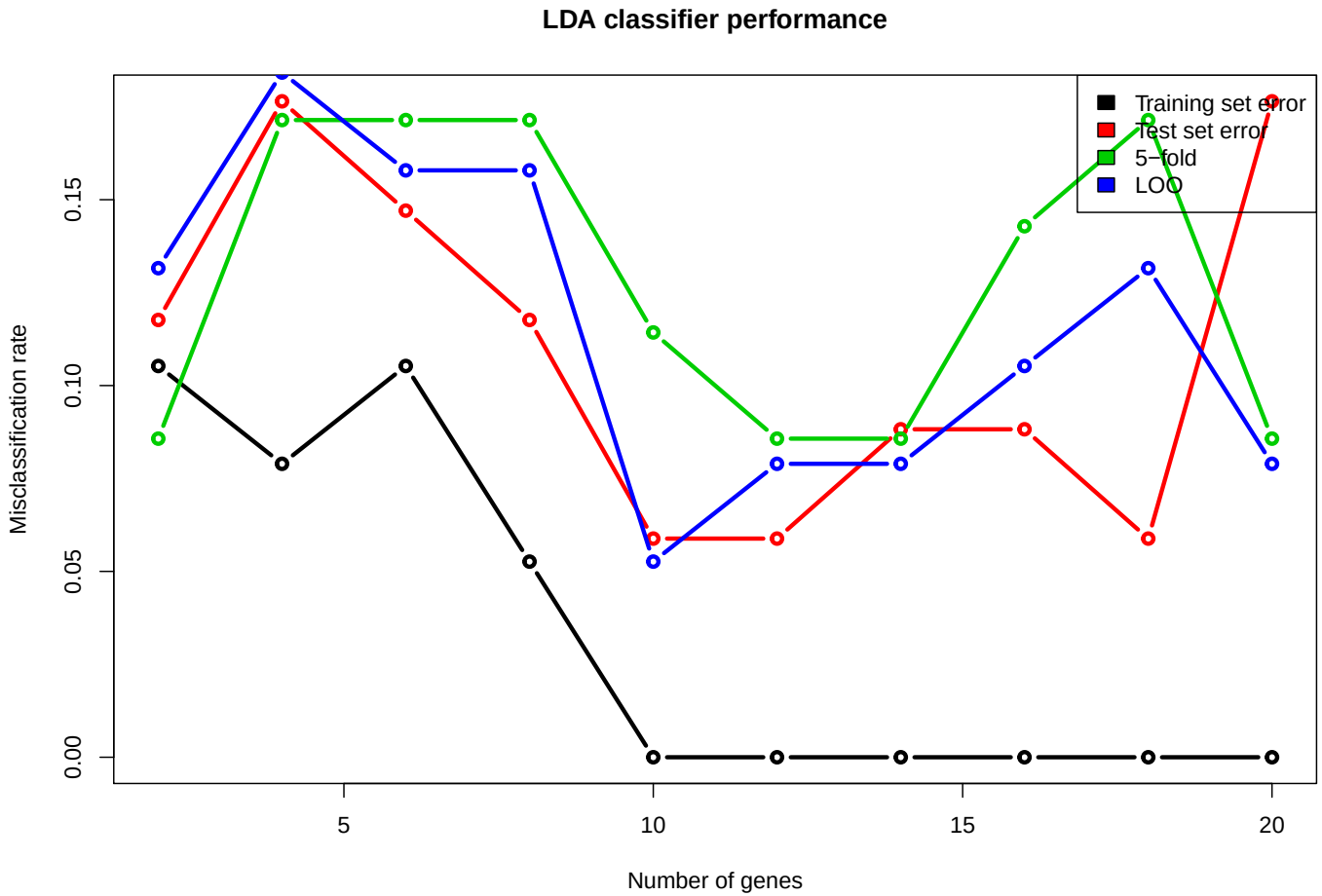
1. Find/Retrieve the most variable genes.
2. Use the top 2, 4, . . . , 20 genes to classify leukemia samples with LDA.
3. Evaluate the performance using a 5-fold CV and LOO CV procedure on the *training set only*.
4. Plot the 5-fold CV and LOO CV error as a function of the number of genes alongside the test-set error rate.

Comment on the relative advantages of each procedure

Solution: compute the CV error for the Leukemia dataset

```
V <- 5
N <- NCOL(golub)
n <- floor(N/V)
fold <- sample(as.vector(mapply(rep, 1:V, n)))

cver <- looer <- numeric(length(ngenes))
for(i in seq_along(ngenes)) {
  cver[i] <- mean(sapply(seq_len(V), function(v) {
    testSet <- data.frame(pheno=pheno[which(fold==v)],
                          t(golub[names(vars)[seq_len(ngenes[i])],
                                  which(fold==v)]))
    trainSet <- data.frame(pheno=pheno[which(fold!=v)],
                           t(golub[names(vars)[seq_len(ngenes[i])],
                                   which(fold!=v)]))

    fit <- lda(pheno~., data=trainSet)
    pred <- predict(fit, testSet)

    er <- sum(pred$class != testSet$pheno)/length(testSet$pheno)
    return(er)
  }))

  looer[i] <- mean(sapply(seq_len(N), function(v) {
    testSet <- data.frame(pheno=pheno[v],
                          t(golub[names(vars)[seq_len(ngenes[i])], v]))
    trainSet <- data.frame(pheno=pheno[-v],
                           t(golub[names(vars)[seq_len(ngenes[i])], -v]))

    fit <- lda(pheno~., data=trainSet)
    pred <- predict(fit, testSet)

    er <- sum(pred$class != testSet$pheno)/length(testSet$pheno)
    return(er)
  }))
```

Solution: compute the CV error for the Leukemia dataset



Classification: CV, training and validation

Typically, classification algorithms have *many tuning parameters* (e.g., number of genes in the previous example).

The usual procedure is to use cross-validation to *tune the parameters* (e.g., select the optimal number of genes) and a *genuine validation set* to compute the final classification performance.

Clustering

Cluster analysis

A common statistical inference problem is to *assign observational units to classes* on the basis of explanatory variables describing these observations.

In *cluster analysis* the *classes are unknown* a priori and need to be "discovered" or "learned" from a learning set of *unlabeled observations*.
A.k.a. class discovery, unsupervised learning, unsupervised pattern recognition.

Cluster analysis

Clustering (unsupervised) is in some sense a more difficult problem than class prediction (supervised). Typically, the issues that must be addressed for class prediction must also be addressed for clustering. In addition, one encounters the following complexities.

- The (number of) *classes are unknown a priori*, i.e., there is no learning set of labeled observations.
- Implicitly, one must have already selected the relevant *observational units/explanatory variables*, *standardization*, and *distance* measures.
- Many algorithms that are appealing from an intuitive or theoretical point of view are *computationally complex*.
- Often, only approximate solutions are available and reproducibility is an issue.

The goals can be *vague*: "Find some interesting and important clusters in my data."

Distances

Inherent in cluster analysis is a notion of *similarity* or *distance between observations*.

Let $d : \mathbb{R}^J \times \mathbb{R}^J \to \mathbb{R}^+$ denote a *distance function* for pairs of J-dimensional real vectors and let $D = (d(X_i, X_j) : i, j = 1, \ldots, n)$ denote the corresponding $n \times n$ *pairwise distance matrix* for the n observations to be clustered.
E.g. Euclidean, one-minus-correlation, ...

R software: `dist` function, `stats` package.

Distances

Given a distance measure between individual observations, one must often also define a measure of *distance between clusters of observations*. There are a number of ways of measuring the *distance between two clusters*.

- *Single linkage.* The distance between two clusters is the *minimum* of all pairwise distances between the members of the two clusters.
- *Average linkage.* The distance between two clusters is the *average* of all pairwise distances between the members of the two clusters.
- *Complete linkage.* The distance between two clusters is the *maximum* of all pairwise distances between the members of the two clusters.
- *Centroid distance.* The distance between two clusters is the *distance between their centroids*, where the definition of *centroid* generally depends on the clustering method, e.g., average, median, medoid.

Clustering procedures

We distinguish between the following two broad types of clustering procedures.

- *Hierarchical methods* provide a hierarchy of clusters, from the smallest set, where all observations are in one cluster, through to the largest set, where each observation is in its own cluster.
- *Partitioning methods* partition the observations into disjoint clusters and usually require specification of the number of clusters.

R software

There are many R and Bioconductor packages dedicated to cluster analysis.

Here, we will use the functions in the `cluster` package (`pam`) and in the `stats` package (`hclust`, `kmeans`).

Elizabeth Purdom and I developed a Bioconductor package, `clusterExperiment` specifically designed for clustering (single-cell) RNA-seq data.

Partitioning methods

The observations are partitioned into *mutually exclusive and exhaustive clusters*.

The *number of clusters* K usually needs to be *prespecified*.

Partitioning clustering algorithms often involve *optimizing some criterion/objective function* by iteratively reallocating observations to clusters, e.g., minimize within-cluster sum-of-squares or sum of distances from nearest medoid.

Partitioning methods: Examples

- *k-means* and extensions to *fuzzy k-means*: Observations assigned to cluster with the nearest mean, serving as cluster prototype (cf. Gaussian mixtures, principal component analysis).
- *Partitioning around medoids* (PAM) (Kaufman and Rousseeuw 1990).
- *Model-based clustering*, e.g., based on Gaussian mixture models (Fraley and Raftery 2002) (McLachlan, Bean, and Peel 2002).

PAM: Partitioning around medoids

Partitioning around medoids (PAM) of Kaufman and Rousseeuw (1990) is a partitioning clustering method which operates on an $n \times n$ *distance matrix*, $D = (d(X_i, X_j))$, of pairwise distances between the n observations to be clustered.

For a *prespecified number of clusters* K, the PAM procedure involves searching for K *medoids*, or representative observations, among the n observations to be clustered.

Given a set of K medoids, the K *clusters* are constructed by assigning each observation to the cluster with the nearest medoid.

PAM can be applied to *any distance* matrix and can therefore accommodate general data types. It tends to be more *robust* than k-means.

PAM: Partitioning around medoids

The goal is to find a set of K medoids, $\mathcal{M}_K = \{M_k : k = 1, \ldots, K\} \subseteq \{X_i : i = 1, \ldots, n\}$, which minimizes the *sum of the distances of each observation to its nearest medoid*.

Specifically, the *optimal set of medoids* $\mathcal{M}_K$ is defined as

$$\mathcal{M}_K \equiv \arg\min_{\{\mathcal{M}:\mathcal{M}\subseteq\{X_i:i=1,\ldots,n\},\,|\mathcal{M}|=K\}} \sum_{i=1}^{n} \min_{M\in\mathcal{M}} d(X_i, M). \tag{9}$$

That is, the PAM algorithm seeks to minimize the *objective function* $f : \mathcal{M} \to \mathbb{R}$ defined by

$$f(\mathcal{M}) \equiv \sum_{i=1}^{n} \min_{M\in\mathcal{M}} d(X_i, M), \tag{10}$$

where $\mathcal{M} \subseteq \{X_i : i = 1, \ldots, n\}$ and $|\mathcal{M}| = K$.

PAM: Partitioning around medoids

For n observations to be partitioned into K clusters, there are $\binom{n}{K} = \frac{n!}{K!(n-K)!}$ possible choices for the K medoids. In practice, it is therefore infeasible to perform an exhaustive search over all sets of K medoids.

The PAM algorithm first seeks a good *initial set of medoids* (`build` phase). It then finds a *local optimum* for the objective function f, that is, a set of medoids such that there is no single switch of an observation with a medoid that decreases the objective function (`swap` phase).

Silhouette width

The *silhouette width* of an observation i is defined as

$$sil(i) \equiv \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1], \tag{11}$$

where $a(i)$ denotes the average distance between the ith observation and all other observations in the cluster to which i belongs and $b(i)$ denotes the minimum average distance between the ith observation and observations in other clusters.

Rousseeuw (1987) suggests using *silhouette plots* to: (i) *select the number of clusters* K; (ii) *assess how well individual observations are clustered*.

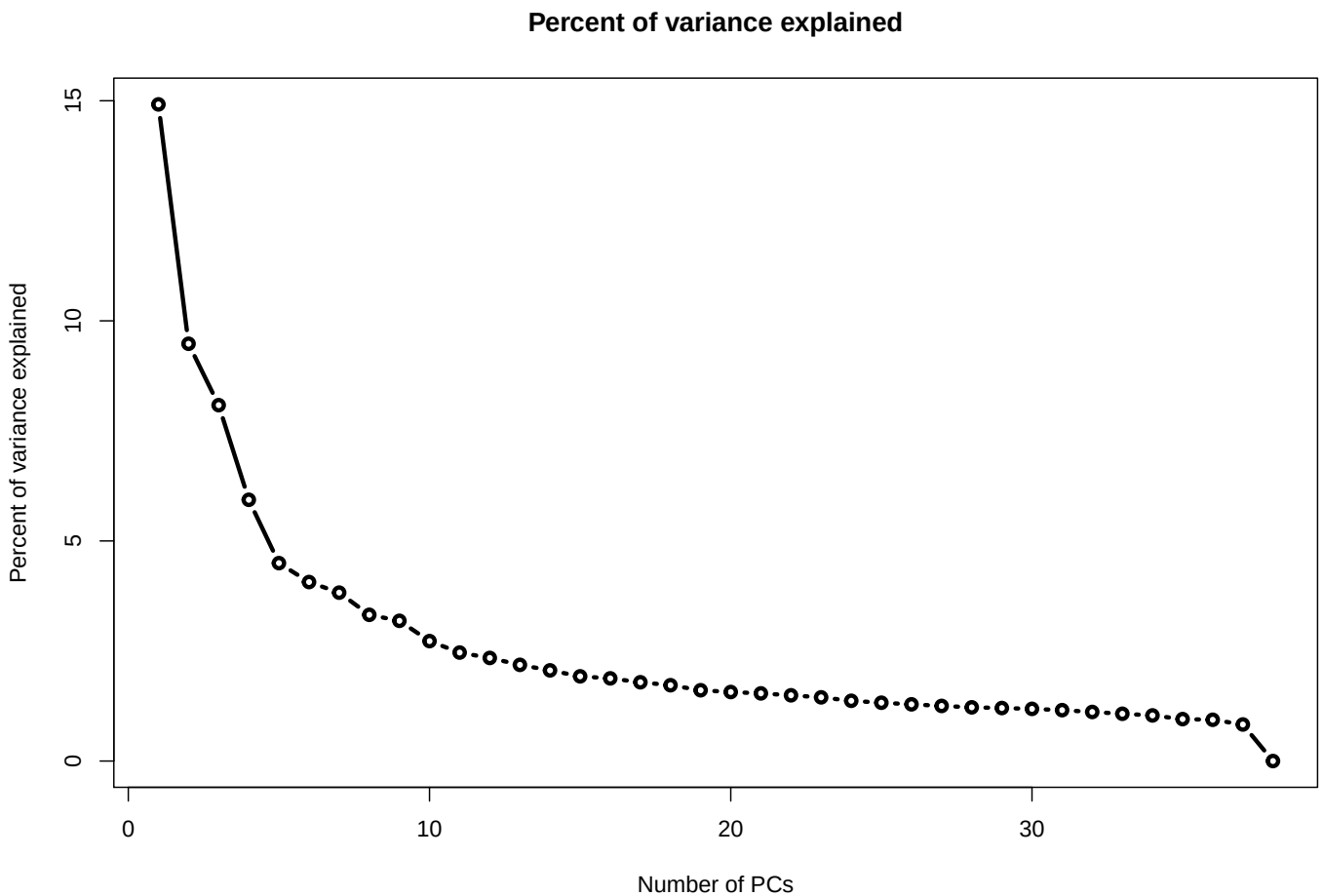
Intuitively, the *larger the silhouette widths*, the *better the clustering*.

Exercise: Cluster the leukemia dataset with PAM

1. Select a suitable number of principal components (hint: `screeplot`).
2. Apply PAM with `k=2` and `k=3` in the space of the selected principal components using 1 - correlation distance (hint: `cor`, `as.dist`).
3. Compare the inferred clusters with the known classification (AML, B-cell ALL, T-cell ALL).
4. Use the average silhouette width to select the "optimal" k (hint: `silinfo` component of `pam` output).

Solution: Cluster the leukemia dataset with PAM

```
pca <- prcomp(t(golub))

plot(pca$sdev^2/sum(pca$sdev^2)*100, xlab="Number of PCs", ylab="Percent of variance explained", type='b',
```




```
p <- 5
```

Solution: Cluster the leukemia dataset with PAM

```r
library(cluster)
r <- cor(t(pca$x[,1:p]))
dd <- 1 - r
dimnames(dd) <- list(as.character(pheno),as.character(pheno))
d <- as.dist(dd)

# PAM, K=2
pam2 <- pam(d, k=2)

# PAM, K=3
pam3 <- pam(d, k=3)

table(pam2$clustering, pheno)
table(pam3$clustering, pheno)

# Graphical summaries
clusplot(dd, pam3$clustering, labels=3, col.p=1,
         col.txt=colors[pheno], main="Bivariate cluster plot")

plot(pam2, which.plots = 2, main="Average silhouette width")
plot(pam3, which.plots = 2, main="Average silhouette width")

# Select best k
K <- 2:10
avgSil <- rep(NA, length(K))
names(avgSil) <- K
for(k in K) {
  avgSil[k-1] <- pam(d, k=k)$silinfo$avg.width
}

# Graphical summaries
barplot(avgSil, names.arg=K, xlab="Number of clusters, K", ylab="Average silhouette width")
plot(K, avgSil, pch=16, cex=2, xlab="Number of clusters, K", ylab="Average silhouette width")
```

Hierarchical clustering

Hierarchical clustering methods produce a *hierarchy* or *tree* of *nested clusters*.

Unlike partitioning methods, they *do not require a prespecified number of clusters* K.

Partitions into clusters can be obtained by *"cutting" the tree into "branches"*, by specifying either a desired number of clusters or cut height.

We distinguish between

- *agglomerative*, i.e., bottom-up, hierarchical clustering;
- *divisive*, i.e., top-down, hierarchical clustering.

Hierarchical clustering: Agglomerative

Start with each observation in its own cluster, i.e., $K = n$ clusters.

At each step, *merge the two nearest clusters* using a measure of *between-cluster distance*.

R software: `hclust` function, `stats` package.

- *Advantages*: computational simplicity.
- *Disadvantages*: cannot correct for early errors.

Hierarchical clustering: Divisive

Start with all observations in the same cluster, i.e., $K = 1$ cluster.

At each step, *split clusters* into two (or more) clusters.

Example: DIvisive ANAlysis (DIANA), `diana` R function in package `cluster`.

- *Advantages*: capture the main structure of the data.
- *Disadvantages*: cannot correct for early errors.

Dendrograms

Dendrograms are often used to visualize the nested sequence of clusters resulting from hierarchical clustering.

While dendrograms are appealing because of their apparent ease of interpretation, they *can be misleading*.

Firstly, the dendrogram corresponding to a given hierarchical clustering is *not unique*. For each merge/split, one needs to specify which subtree or branch should go on the left and which on the right.
For example, in agglomerative hierarchical clustering, there are $2^{(n-1)}$ possible dendrograms.
The default in the R function `hclust` is to order the subtrees so that the tighter cluster is on the left.

Dendrograms

A second and perhaps less recognized shortcoming of dendrograms is that they *impose structure* on the data, instead of revealing structure in these data.

A dendrogram is a valid graphical representation of data only to the extent that the pairwise distances between observations possess the hierarchical structure imposed by the clustering procedure.

Exercise: Cluster the leukemia dataset with `hclust`

1. Use the same distance computed for the PAM analysis.
2. Apply `hclust` with an appropriate agglomeration distance.
3. Visualize the results with a dendrogram (hint: `plot`)
4. Cut the tree to obtain three clusters (hint: `cutree`)
5. Compare the inferred clusters with the known classification (AML, B-cell ALL, T-cell ALL).

Solution: Cluster the leukemia dataset with `hclust`

```
hc <- hclust(d, method="average")

table(cutree(hc, 3), pheno)

plot(hc, main="Hierarchical clustering dendrogram", sub="Average linkage agglomeration, one-minus-correlat
rect.hclust(hc, k=3, border=colors)
```

Resampling-based and Ensemble clustering

As with classification, many clustering algorithms have been proposed in the statistical literature.

Some general features that improve the *robustness* and *stability* of the clustering procedures are:

- *Resampling-based clustering*
- *Sequential clustering*
- *Ensemble / consensus clustering*

These ideas may also help in selecting the number of clusters.

Resampling-based clustering

Given an underlying clustering strategy, e.g., k-means or PAM with a particular choice of k, we can:

1. Subsample the data, e.g. 70% of samples.
2. Find clusters on the subsample.
3. Create a co-clustering matrix D:
    - % of subsamples where samples were in the same cluster.

Note that Step 3. is itself a clustering step, and hierarchical clustering may be used.

Sequential clustering

Initially proposed by Tseng and Wong (2005) to cluster genes in microarray data.

Given an underlying clustering strategy, e.g., k-means or PAM, we can:

1. Range over k (optionally using the subsampling strategy).
2. The cluster that remains stable across values of k is identified and removed.
3. Repeat until no more stable clusters are found.

Ensemble clustering

A.k.a. *consensus clustering*. Given an underlying clustering strategy, e.g., k-means or PAM, we can

1. Run the algorithm *varying the tuning parameters* (e.g., number of variables p or number of clusters k).
2. Collect the results from multiple clustering algorithms.
3. Compute a consensus by, e.g., majority vote.

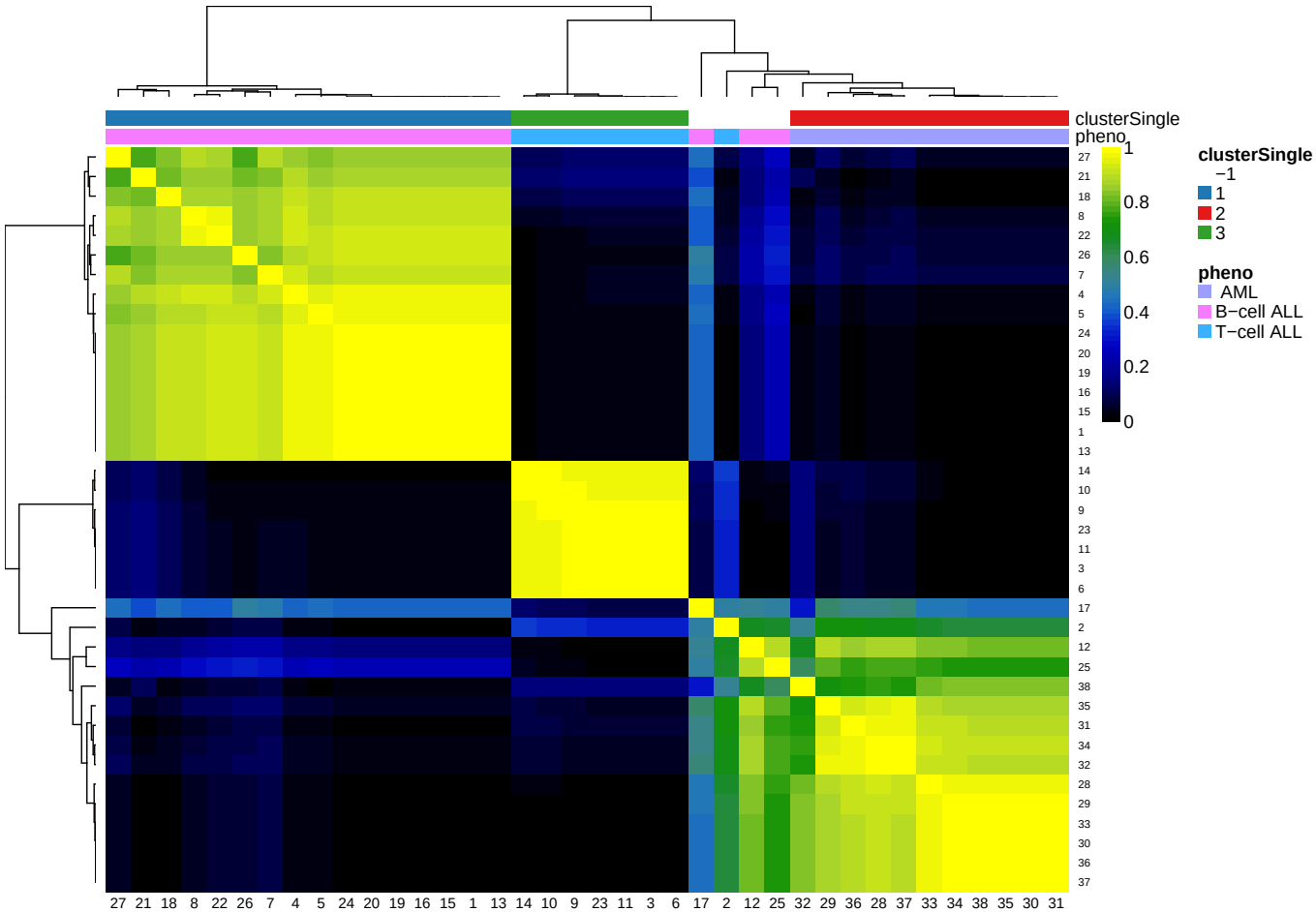
As often happens in statistics, *averaging* leads to more prediction power.

Example: Resampling

```
library(clusterExperiment)

cl <- clusterSingle(golub[names(vars)[1:500],],
                    subsample = TRUE, sequential = FALSE,
                    clusterFunction = c("hierarchical01"),
                    clusterDArgs = list(minSize=5, alpha=0.3),
                    subsampleArgs = list("k"=3,"clusterFunction"="pam"),
                    dimReduce = c("PCA"), ndims=10)
```

Example: Resampling

```
colData(cl) <- DataFrame(pheno=pheno)
plotCoClustering(cl, sampleData="pheno")
```



Example: Sequential

```
cl2 <- clusterSingle(golub[names(vars)[1:500],],
                     subsample = TRUE, sequential = TRUE,
                     clusterFunction = c("hierarchical01"),
                     clusterDArgs = list(minSize=5, alpha=0.3),
                     subsampleArgs = list("clusterFunction"="pam"),
                     seqArgs = list(k0=5, remain.n=10, top.can=5),
                     dimReduce = c("PCA"), ndims=10)
```

```
## Number of points: 38     Dimension: 10
## Looking for cluster 1 ...
## k = 5
## k = 6
## Did not find 5 clusters: found 3,2 clusters for k= 5,6 , respectively
## Cluster 1 found. Cluster size: 7     Remaining number of points: 31
##  Looking for cluster 2 ...
## k = 4
## k = 5
## Did not find 5 clusters: found 3,2 clusters for k= 4,5 , respectively
## Cluster 2 found. Cluster size: 6     Remaining number of points: 25
##  Looking for cluster 3 ...
## k = 3
## k = 4
## Did not find 5 clusters: found 2,2 clusters for k= 3,4 , respectively
## Cluster 3 found. Cluster size: 5     Remaining number of points: 20
##  Looking for cluster 4 ...
## k = 3
## k = 4
## Did not find 5 clusters: found 2,1 clusters for k= 3,4 , respectively
## Cluster 4 found. Cluster size: 5     Remaining number of points: 15
##  Looking for cluster 5 ...
## k = 3
## k = 4
## Found  1,0 clusters for k= 3,4 , respectively. Stopping iterating because zero-length cluster.
```

Example: Ensemble

```
clusterMany(cl,
            dimReduce="PCA",  nPCADims=c(2, 3, 5),
            ks = 3, clusterFunction = "pam") %>%
  combineMany %>%
  plotClusters(sampleData="pheno")
```

Reproducible research

Additional resources

- Data 8: Foundations of Data Science: data8.org
- Karl Broman's tutorials: kbroman.org/pages/tutorials.html
- R markdown and knitr tutorial: github.com/ijlyttle/user2016_knitr
- The Elements of Statistical Learning (book by Hastie, Tibshirani, Friedman): statweb.stanford.edu/~tibs/ElemStatLearn/

References

Breiman, L. 1996. "Bagging Predictors." *Machine Learning* 24: 123–40.

———. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.

Breiman, L., J. H. Friedman, R. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Boca Raton, FL: Chapman & Hall/CRC.

Dudoit, S., and J. Fridlyand. 2003. "Bagging to Improve the Accuracy of a Clustering Procedure." *Bioinformatics* 19 (9): 1090–9. `http://bioinformatics.oxfordjournals.org/content/19/9/1090.abstract`.

Fix, E., and J. Hodges. 1951. "Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties." Randolph Field, Texas: USAF School of Aviation Medicine.

Fraley, C., and A. Raftery. 2002. "Model-Based Clustering, Discriminant Analysis, and Density Estimation." *Jasa* 97 (458): 611–31.

Friedman, J., T. Hastie, and R. Tibshirani. 2000. "Additive Logistic Regression: A Statistical View of Boosting." *Annals of Statistics* 28: 337–404.

Golub, Todd R, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, et al. 1999. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring." *Science* 286 (5439). American Association for the Advancement of Science: 531–37.

Hastie, T., R. Tibshirani, and J. H. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Kaufman, L., and P. J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley.

Laan, M. J. van der, E. C. Polley, and A. E. Hubbard. 2007. "Super Learner." *Sagmb* 6 (1): Article 25.

Mardia, K. V., J. T. Kent, and J. M. Bibby. 1979. *Multivariate Analysis*. London, New York: Academic Press.

McLachlan, G. J., R. Bean, and D. Peel. 2002. "A Mixture Model-Based Approach to the Clustering of Microarray Expression Data." *Bioinformatics* 18: 413–22.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rousseeuw, P. J. 1987. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Journal of Computational and Applied Mathematics* 20: 53–65.

Tseng, George C, and Wing H Wong. 2005. "Tight Clustering: A Resampling-Based Approach for Identifying Stable and Tight Patterns in Data." *Biometrics* 61 (1). Wiley Online Library: 10–16.

Vapnik, V. 1998. *Statistical Learning Theory*. 1st ed. New York: Wiley.