**Data Analysis & Visualization in R**

# I. Introduction to R

Exam:

* Long, hard to finish 🡪 grades will be adapted
* Will not simulate data in exam

Script:

* Experiment in console
* Exiting R studio without saving: temporary variables are lost (I want this)

\_\_\_\_\_\_\_\_\_\_\_

* Logical Vector = true or false
* Indices start at 1
* Data Science = making data-driven observations to formulate testable hypotheses

**R code Basics:**

* ls() -> see variables saved in workspace
* sqrt(a)
* log(a)
* help shortcurt: ?log
* args(log) 🡪 quick look at arguments without opening help system
* data() 🡪 see all available datasets
* pi, Inf, co2 🡪 prebuilt

Variable names: start with letter, no spaces, not predefined

Rows = **Observations;** Columns = **variables**

**Data Frame:** combining **different data types**; special case of lists

* class(a) 🡪 type of the object
* str(a) 🡪 structure of an object
* head(a) 🡪 first 6 lines
* data frame accessor: a$column
* names(df) 🡪 variable names

**Vectors**: several entries of **same type**

**Factors**: storing **categorical data**

* levels(…) 🡪 default: alphabetical; reorder() changing order of levels; as.factor()
* is.na(xy)
* length(…)
* as.integer() or by adding L: 1L 🡪 class(1L) = integer
* as.numeric()
* as.character()
* == 🡪 test for equality; != inequality

**Lists:** storing any combination of **different types**:

* + record <- **list**(name = "John Doe",

*variable name:* student\_id = 1234,

* + grades = **c**(95, 82, 91, 97, 93),
  + final\_grade = "A")
  + record[["student\_id"]] *## 1234 accessing the value == [[2]]*

**Matrices:** entries have to be all the **same type**

* mat <- **matrix**(1:12, 4, 3)
  + - values, rows, columns
* mat[2, 3] # second row, 3rd column = 10
  + **returns a vector**
* good for matrix algebra operations
* Entire 2nd row (same for data frame): mat[2, ]
* Entire 3rd column: mat[, 3]
* Subsetting 2nd & 3rd column: mat[, 2:3]
* Converting matrices to data frame: as.data.frame(mat)

**Creating Vectors**

* One vector with several entries of the **same type**
* vector <- c(…,…,…)
* vector <- c(name = .., name = …)
* names(vector)
  + codes <- **c**(380, 124, 818)
  + country <- **c**("italy","canada","egypt")
  + **names**(codes) <- country
* *vector* x 10 🡪 apply computation on every entry

**Sequences**

* seq(1,10,1)
* *start, end (included), increment*
* class(1:10) = “integer”; class(seq(1,10,0.5)) = “numeric”

**Subsetting 🡪 same for data.table**

* vector[c(1,3)] = access of 1st and 3rd entry
* vector[1:2] = access first 2 elements
* vector[“name”]
* vector[c(“name1”, “name4”)]

**Coercing**: R coerces automatically if you mix data types in vector

* can introduce **NAs**

**Sorting**

* sort(…) 🡪 sorts vector in increasing order; changed **sequence**
* order(…,…) 🡪 returns vector of **indices** that sorts the input vector (“-“ for dec)
  + index <- **order**(x) # or order(dframe$column)
  + x[index] # lowest to highest

**Max/ Min**

* max(dframe$column)
* which.max(dframe$column) 🡪 index of largest value
  + i\_max <- **which.max**(murders$total)
  + murders$state[i\_max]

**Rank**

* rank(df, ties.method = “max”) 🡪 ranking entries from small (1) to large

**Recycling**: unmatched lengths in operations: Warning but notstrsplit error; R computes it

**Computations:** Operations on vectors occur *element-wise*

* murder\_rate <- murders$total / murders$population \* 100000
* murders$abb[**order**(murder\_rate)]
* **ind** <- murder\_rate <= 0.71 *## returns logical vector with true*
* murders$state[ind]
* **sum**(ind) # =5 🡪 True = 1, False = 0

**Logical Operations**

* west <- murders$region == "West"
* safe <- murder\_rate <= 1
* **ind** <- safe & west
* murders$state[ind]
* **which**: indexes which entries of a logical vector are TRUE
  + ind <- **which**(murders$state == "California")
  + murder\_rate[ind]
* **match**: which indexes of a second vector match each of the entries of the first
  + ind <- **match**(**c**("New York", "Florida", "Texas"), murders$state)
  + murder\_rate[ind] # murder rates of these states
* **%in%:** TRUE or FALSE vector whether the elements are in the vector:
  + **c**("Boston", "Dakota", "Washington") %in% murders$state
    - *nur 3 Elemente lang da dirket selektiert*

**match**(**c**("New York", "Florida", "Texas"), murders$state)

…same index vector (different order)…

**which**(murders$state%in%**c**("New York", "Florida", "Texas"))

# II. Data Wrangling

**data.table**

* faster than df: Operates columns **by** **reference**
  + R is not copying the data
    - because overwriting the old data.table!
      * new\_dt <- old\_dt 🡪 new object created, both point to same memory 🡪 old\_dt will be affected by changes
    - -> Create a copy() to avoid this 🡪 changes of the copy do not affect the old one
* DT[ i, j, by ] #
* | | |
* | | -------> grouped by what?
* | -------> what to do with the columns? Computations
* ---> on which rows?

**Creating a data.table:**

* DT <- **data.table**(x = **rep**(**c**("a","b","c"), each = 3), y = **c**(1, 3, 6), v = 1:9)
* as.data.table(df, keep.rownames = F) 🡪 converting any object to dt

**Reading and inspecting data.table:**

* fread(“./………. .csv”)
* head(dt)
* ncol(dt)
* nrow(dt)
* dim(dt) 🡪 returns nrow and ncol
* summary(dt) 🡪 basic statistics on columns
* dt[, unique(column)] 🡪 investigating categorical columns
* dt[, table(column)] 🡪 how often each category occurs
  + table takes one or multiple vectors and returns the frequency of each element

**Subsetting**

* dt[c(x,y)] 🡪 not consecutive accessing
* dt[column == “…”]
* dt[column %in% c(“..”,”..”)
* Concatenating multiple conditions with | (or) & (and)
* dt[, c(colA,colB)] 🡪 accessing multiple columns, returning as vector
* dt[, .(colA , colB)] 🡪 (list) accessing multiple columns, returning as data.table

**Column Operations**

* access columns by name
* dt[, mean(.., na.rm = T), by = … ]
  + by 🡪 executes j command by groups
* Multiple: dt[, .(newname = mean(.., na.rm = T), newName2 = median(..,na.rm = T)]
* dt[…, colA / colB]
* sapply(dt, function to be applied on each element of dt)
* Counting occurrences = .N
  + Dt[, .N] == nrow(dt)
  + # of flights per airline: flights[, .N, by = “Airline”]
  + # of flights arriving in JFK: flights[ destination == “JFK”, .N, by = “Airline”]
* Creating new columns: :=
  + **Multiple** new columns at once:
    - *dt[, `:=` (Sepal.Area = Sepal.Length \* Sepal.Width,*
    - *Petal.Area = Petal.Length \* Petal.Width)][1:3]*
  + Removing: := NULL
    - books\_dt[, **c**("Image\_URL\_S", "Image\_URL\_M", "Image\_URL\_L")**:=**NULL]

Can **concatenate data.table operations** with **[][][]**

Pipe operator: needs **.** to where the argument goes

# III. Tidy Data

Tidy =

1. Each **variable** has its own **column**.
2. Each **observation** has its own **row**.
3. Each **value** has its own **cell**.

🡪 easier: manipulate, vectorized operations, plotting function, hypotheses testing…

Common signs of untidy datasets:

* **Column headers are values**, not variable names
* **Multiple variables** are stored **in one column**
* **Variables** are stored in **both rows and columns**
* A **single observational unit is stored in multiple tables**

## **Melting, Casting, Separating, Uniting 🡪 tidyr**

Melting: **From Wide to Long**

**melt**(dt, id.vars, measure.vars, variable.name, value.name)

* id.vars = key of the rows (stays the same)
* measure.vars = which variables(columns) shall be melted
  + need only one of those 2
* variable.name = new column name for the melted variables
* value.name = column name of values

Casting: **From Long to Wide**

**dcast**(dt, ... ~ colname, value.var = "value")

* Separating multiple values to multiple columns
* ~ colname = col which becomes wide: categories out of this column
* value.var = which column the values are extracted from
* **…** 🡪 everything else
* ~ 🡪 **as a function of**

Separating Columns

**separate**(dt, col =… , into = **c**("…", " … "), *extra = “drop”*)

* Splitting one column into new columns

Uniting Columns

**unite**(dt, col = …, oldCol, oldCol, sep = " ")

* col = new column name, old column, old column
* sep = separating characters used to unite

S.31: melting days, uniting over month and date

**Working on Strings:**

* Splitting column in 2
  + **tstrsplit**(LearningPlatformSelect, ',')[]
* Replacing all occurrences of a pattern:
  + **gsub**(„oldPattern“, „replacement“, data)
    - weather\_dt[, day := gsub('d', "", day)] #overwriting column day
* Concatenating vectors: paste (..., … ,…, sep = " ")
* Converting upper case to lower case: tolower(element)

**Concatenating** **tables**

* all file names int a **list**:
  + files <- **list.files**('path\_to\_your\_directory', full.names = T)
  + lapply(*on what we apply something*, the *actual* *function*)
    - returns list of data.tables
    - *# name the list elements by the filenames*
    - **names**(files) <- **basename**(files)
    - *# read all files at once into a list of data.tables*
    - tables <- **lapply**(files, fread)
  + accessing the first table in a list: list[[1]] *here: tables*
* Bind all tables into one with *id column*: list name (filename) as id column
  + dt <- **rbindlist**(tables, idcol = „filename“)

## Merging tables = Joins

**merge**(

x, y, *# tables to merge*

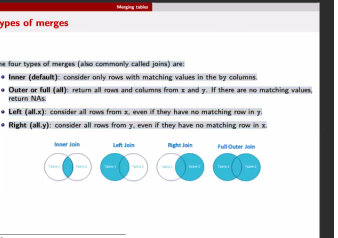
by = NULL, by.x = NULL, by.y = NULL, *# by which columns*

all = FALSE, all.x = all, all.y = all) *# types of merge*

* **Inner** = Default 🡪 only matching values 🡪 **by** = …, **all = FALSE**

🡪 keep original order: merge(… , sort = False) am Ende

* **Outer** or full (all) 🡪 all values, **all = TRUE**, inserts NA for no matching values
* **Left** (all.x) 🡪 all from left, no matchings in y: NA🡪 all.x = TRUE
* **Right** (all.y) 🡪 all from right, no matchings in x: NA 🡪 all.y = TRUE



Merging by several columns:

* merge(dt1, dt2, by=**c**("firstname", "lastname"))
* column names need to be named distinctly
  + 🡪 R produced when merging colname.x & colname.y

- depending on your downstream analysis (focus), you can have a tidy representation

Database **Back-Ends**: Maintaining a database

* Normalized representations: no multiple types of observational units are stored in same table
  + facilitates maintenance of the data consistency by reducing redundancy
* *Entering* data still in back-end

**Data Analysis (R) is Front-end:** Ready.to-use format

* No normalization needed 🡪 we want to visualize and work on data, more practical to have mergers and have some redundant information for better analytics and visualization
* merge table useful and common denominator of multiple analysis
* Want to combine multiple tables, initially separated in the back-end, even if information is replicated
* Always want column denominator
* On front end, don’t need normalized data

🡪 Choice of representation (normalized or not) depends on context (back or front-end)

# IV. Low Dimension Visualization

* Scientific Method: Observation, Question, Hypothesis, Experiment, Analysis, Conclusion
* **Summary statistics lose information**
* Plotting allows to:
  + Facilitates making new observations
  + Facilitates communicating findings
  + Helps debugging code or identifying bugs In the data (wrong entries, outliers)

**Grammar of graphics:** Visualization theory: makes code reusable and more maintainable

* + - * Separate data from aesthetics
      * Definition of common plot elements
      * Composition of these common elements (combine elements as **layers**)

**🡪** *Always* need 3 layers:

1. **Data**
2. **Aesthetics**: Map variables to graphical elements
3. **Geometric** Objects

🡪 *Often used*:

1. **Scales**: f.e. applying log-scales
2. **Facets**: splitting data f.e. for 2 years 🡪 facet\_grid(**~**year) / facet\_wrap(~..) 🡪 “by year”

*🡪 Useful, but with care:*

1. **Stats**: statistical transformations: counts, means, medians, regression lines, smooth trends 🡪 Only do this when you understand it good

🡪 *Domain-specific*

1. Coordinate system

* Ggplot(data, aes(x,y)) <- In there is the data layer
* Aesthetics maps the variables to visualization: axis, colors, point shapes, etc.
* + geom\_point() 🡪 Geometric layer
* ggplot returns an object which can be stored
* saveRDS(plotobject, “pathtoplot.rds”)
* scale\_x\_log10()
* labs(x=..,y=…,)
* … color = … ) 🡪 in aes, Mapping variables to colors
* … shape = … ) 🡪 in aes, Mapping variables to shapes
* … size = … ) 🡪 in aes, Mapping variables to sizes of points
* Don’t use labels only when your absolutely finished
* Creating themes as a uniform style across document and add this with + mytheme
* Global mapping of aes() in ggplot() 🡪 is **inherited** to all layers
* Local mapping prevents other layers to have access to the aesthetics mapping
  + Ggplot(data) + geom\_point(aes(x,y)) + stat\_smooth()
    - 🡪 doesn’t work, as stat doesn’t know aes(x,y)
  + 🡪 common to have aes(x,y) in ggplot() globally

Example:

**ggplot**(data=gm\_dt, **aes**(x=gdpPercap, y=lifeExp)) +

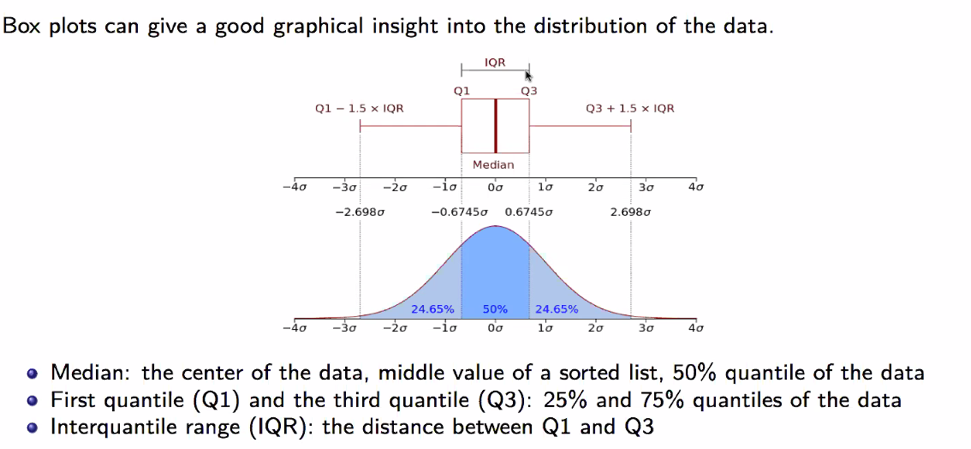
**geom\_point**(**aes**(color=continent, size=pop)) +

**facet\_grid**(~year) + **scale\_x\_log10**() +

**labs**(y="Life expectancy at birth", x="per-capita GDP", size = 'Population') + mytheme

* facet\_grid(~…) / facet\_wrap(~…)

## **1 Continuous variable**

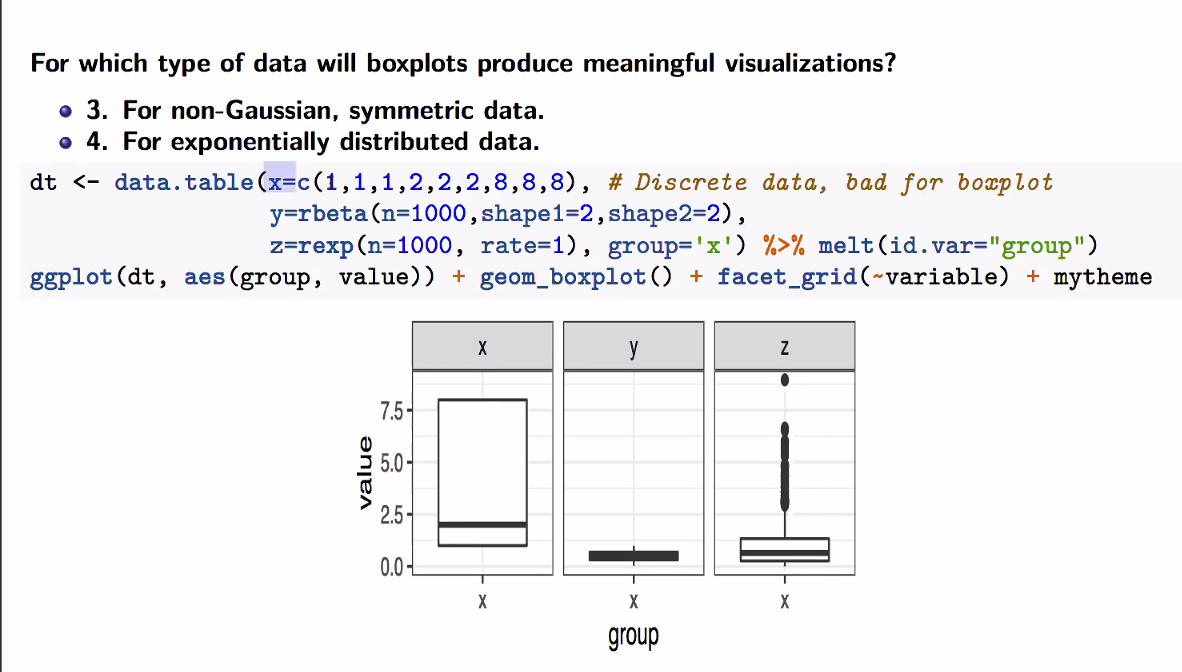
* **Histogram** 🡪 geom\_histogram(bins=…) bins = used for size of buckets
  + Distribution
* **Density plot** 🡪 geom\_density()
  + 🡪 distribution of a numeric variable
* ggtitle(‘title’)
* **Boxplots**
  + More about robust statistics, shows outliers clearly
  + Coord\_flip() to flip boxplot horizontally
  + particularly suited for **plotting non-gaussian symmetric and non-symmetric data** and for plotting **exponentially distributed data**.
  + **Not** well suited for **bimodal data**, they only show one mode (the **median**)
  + Quantiles: quantile(data, 0,25) (oder 0,75)

## **2 variables: one continuous, one discrete**

* **Barplots**: One value per category
  + geom\_bar(stat = ‘identity’) looks up the relevant variables
  + Add Standard deviations to bars
* Adding dots or points to boxplot:
  + geom\_dotplot(binaxis="y", stackdir="center", dotsize=0.3) or geom\_jitter()
* Favor boxplots over barplots for showing median as they show more data
  + 🡪 Increase the data/ink ratio 🡪 barplot is “waste of ink”
* **Alternative to Boxplots:**
  + **Violin Plots** 🡪 shows **entire** **distribution** of data, particularly interesting for **multimodal** **data; violin plot gets wider, the more data points in this region**
    - geom\_violin()
  + Beanplots: geom\_beeswarm()

**General Principles**

* 🡪 Increase the data/ink ratio 🡪 barplot is “waste of ink”
* 🡪 Show data as raw as possible



Bimodal = 2 peaks 🡪 cant see median or mean for 2 peaks

## **2 continuous variables**

* **Scatterplots:** geom\_point()
  + Use annotation\_logticks()
  + Use facet\_wrap(~…) when too many colours, to separate into diff. plots
* Log scaling: scale\_x\_log10(), scale\_y\_log10() 🡪 few points which are notably larger than most of the points
* **Text labling**: geom\_text(aes(label = …)
  + Ggrepel library: geom\_text\_repel(aes(label =…) 🡪 no overlaps
* **Line Graphs**: geom\_line()
  + Continuous variable for x (time)🡪 use line
* **2D-density plots** 🡪 geom\_hex() instead of geom\_point() when huge number of points

------------------------

* **Plot Matrix:**
  + Library(GGally) 🡪 ggpairs(data, colums = c(…)
  + Correlation and distributions of a few variables in a matrix-like representation (=a data.table)
* **Correlation Plot:**
  + Library(GGally) 🡪 ggcorr(data, geom = ‘circle’)
  + graphical representation of correlation matrix

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**You should remember:**

# V. High Dimension Visualization

Scalars = x

Vectors = **x**

Matrixes **= X.** 🡪 Transposed with T

Need to prepare for **data matrix:**

mat <- **as.matrix**(mtcars[1:10, **c**("mpg", "carb", "hp", "wt")])

**rownames**(mat) <- **rownames**(mtcars)[1:10]

**colnames**(mat) <- **c**("Miles.per.gallon", "Carburetor", "Horsepower", "Weight")

**head**(mat) *# A look at the 5 first rows*

## **Heatmaps**

* Data as **Matrix** or **Data.Frame**
* **Define** **rownames**(mat) <- dt[….]
* Beyond 5-10 variables, scatterplots become problematic
* Library(pheatmap)
  + pheatmap(mat, cluster\_rows = F, cluster\_cols=F, scale = “none/column”, clustering\_method = …, annotation\_row = row.ann, show\_rownames = F)
* add annotation row:
  + *label the row names to be able to annotate rows*
  + **rownames**(mat) <- 1**:nrow**(mat)  
    *create a data.frame for the row annotations*
  + row.ann <- **data.table**(colname = data**$**column)
  + *adding results of a clutering method to ann.row after cutting the tree*
  + row.ann[, complete := factor(complete)]
* table(one\_clustering\_method, other\_clustering\_method) 🡪 comparing cluster alloc.

**Centering & scaling variables** (usually for data matrix columns) 🡪 to bring variables to a same scale

* **Centering**: substract mean
* **Scaling**: centering then dividing by standard deviation:

🡪 pheatmap(.., scale = ‘column’) (both done this way)

* Difficult for 1000\*1000 matrixes

**Clustering**

helps finding patterns in data **matrices**

## **K-means clustering**

* aims to partition the observations into K non-overlapping clusters
* **Number of clusters K is predefined**
* Centroids = μ(shown as crosses)
* Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungSum of the squared distances between observations x and centeroids as close as possible 🡪 one aims to determine the clusters C1,...,CK and the centroids μ1,...,μK in order to minimize the within-cluster sum of squares
* Assume know centeroids: Associate every observation to its closest centeroid

K-means algorithm:

1. Choose the K initial centroids (one for each cluster). Different methods such as sampling random observations are available for this task.
2. Assign each observation xi to its nearest centroid by computing the Euclidean distance between each observation to each centroid.
3. Update the centroids μk by taking the **mean value** of all of the observations assigned to each previous centroid.
4. Repeat steps 2 and 3 until the difference between new and former centroids is less than a previously defined threshold.

🡪 At every iteration, and at every step 2 and 3, the within-cluster sum of squares decreases

🡪 no guarantee to reach optimal solution, depends on initialization

Assumptions that need to be met when performing k-means clustering:

* **Number of clusters K is properly selected (has to be known )**
* **In each cluster** the **variables are not correlated** and have **equal variance** = clusters **isotrophically** distributed
* Clusters have equal (or similar) **variance**
* Clusters are of similar **size**

K-means clustering in R

k <- 2

X <- **scale**(mat) *# use the* ***scaled*** *variables for the clustering*

clust\_km <- **kmeans**(X, k, nstart = 20) *# K-means 20 times*

clust\_km$cluster *# clusters of the best clustering*

Update heatmap with results of clustering: use annotation\_row argrument of pheatmap()

*# create the row annotation data frame*

row.ann <- **data.frame**(

kmeans = **paste0**("C",clust\_km$cluster) *# we call the cluster C1,..,CK.*

* *Can add more here -*

)

*# rownames are used to match the matrix rows with the row annotation data frame.*

*# We can now safely reorder the rows of X.*

**rownames**(row.ann) <- **rownames**(X)

*# o: order of the rows to have increasing cluster number*

o <- **order**(clust\_km$cluster)

**pheatmap**(

X[o,], *# X with ordered rows according to cluster number*

scale='none', *# no need to scale, X is scaled*

annotation\_row = row.ann,

cluster\_rows=FALSE *(T when doing HC),* cluster\_cols=FALSE

)

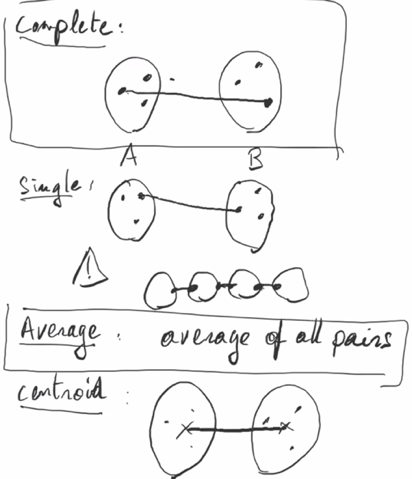
## **Hierarchical Clustering**

🡪 exploring multiple levels of clustering granularity at once by computing nested clusters

🡪 dendrogram

🡪 no objective function like k-means (no predefined number of clusters)

🡪 *linkage rules* define dissimilarity between clusters: definition of *distance* between sets

* **Complete**: distance between sets is made of points furthest away to each other
* **Single**: distance between sets is made of points closest to each other
* **Average**: average distances of all points
* **Centroid**: distance between mean of the sets

Hierarchical clustering in R

d <- **dist**(X) *# compute distance matrix with default (Euclidean)*

hc <- **hclust**(d) *# apply hierarchical clustering with default (complete linkage rule)*

**plot**(hc, hang=-1) *# hang=-1 align observation labels at the bottom of the dendrogram*

* The height in the dendrogram at which two clusters are merged = distance between those two clusters

**Pretty Heatmaps including hierarchical clustering**

**pheatmap**(X, cluster\_rows=**TRUE**, cluster\_cols=FALSE, scale='none')

🡪 most **similar** rows **together**; **outlier** as “furthest” **away**

s

Defining distinct clusters by **cutting the tree**/dendogram at a certain height

* Getting 3 clusters: clust\_hc <- cutree(hc, k=3)
* When using a pheatmap: clust\_hc <- cutree(pheatmap$tree\_row, k=3)
* Adding results to row.ann after cutting:
  + row.ann[, complete := factor(complete)]

**Differences between k-means and HC**

* Time complexity of K-means clustering is linear, HC is quadratic
* K-means: start with random choice of centroids for each cluster
  + Results produced by k-means depend on initialization
  + HC outputs reproducible results
* K-Means: requires number of clusters a priori
  + HC: number of clusters we find appropriate can be found a posteriori by interpreting the dendrogram

Visualizing the outcome of K-means and hierarchical clustering cut for 3 clusters

## **Rand Index:** Comparing clusterings

*Computation*

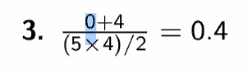
*# PAIR* ***c*** *| same cluster in both methods* ***a*** *| different clusters in both methods* ***b*** 🡪 (a+b) / c

* In R: compute all pairwise rand indices for data table (as.numeric() for all before):
  + rand <- **apply**(row.ann, 2, **function**(i)  
    **apply**(row.ann, 2, **function**(j) **rand.index**(**as.numeric**(i), **as.numeric**(j))))
* Two partitions (from different clustering algorithms) are considered to be similar when many pairs of points are grouped together in both partitions
* Values between 0 and 1 (= all pairs that are in the same cluster in the partition X are also in the same cluster in the partition Y **and** all pairs that are not in the same cluster in X are also not in the same cluster in Y)
  + Two partitions are identical = 1; the higher 🡪 the more similar both partitions
* Ein Bild, das Text enthält.

  Automatisch generierte Beschreibung

Ein Bild, das Text, Whiteboard, weiß enthält.

Automatisch generierte Beschreibung



My explanation of the quiz again for everybody (and how I solved it):

1. List all possible pairs of points (observations). There are 10 of them: {A,B}, {A,C}, {A,D},... {D,E}
2. Analyse each pair to compute quantity a: meaning count the pairs that are in the same cluster for both methods --> 0 pairs in this example
3. Analyse each pair to compute quantity b: count all pairs that are not in the same cluster in both methods: in this case these are the  4 pairs:{A, D}, {A,E}, {B,C}, {B,D}
4. Hence the rand index is (0+4) / 10

## **PCA: Dimensionality reduction**

Principal Component Analysis (Pearson, 1901): projection of the data on the **subspace of dimension q that is closest to the data according to the sums of the squared Euclidean distances.**

* Ein Bild, das Text enthält.

  Automatisch generierte Beschreibungsearch for a line lying as close as possible to the data, in the sense of least squared Euclidean distances 🡪 the perpendicular distance to the line is minimized
* **w = direction vector of the line of length 1 = *first principal component (PC1) 🡪* PC1 maximizes the variance of the projected data**

*Pythagoras’ theorem gives us variances of P(x)*

* **proportion of variance** **captured by PC1** = ratio of the variance of the projected data over the total variance of the data 🡪 between 0 & 1

🡪 The higher it is, the smaller the sum of squared distances, the closer the line is to the data. The proportion of variance quantifies how good our dimension reduction is

**PCA in higher dimensions**

* In general: One searches for **q-dimensional plane** that is **closest to the data in terms of sums of squared Euclidean distances.** Also the q-dimensional plane that **maximizes the variance of the projected data.**
* An important property relates principal components to the eigendecomposition of the covariance matrix.
  + Covariance matrix is symmetric, positive matrix with eigenvectors w ordered by decreasing eigenvalues lambda
  + Result: The PCA q-dimensional plane, i.e. the q-dimensional plane that is closest to the data in terms of sums of squared Euclidean distances, is the **plane spanned by the first q eigenvectors of the covariance matrix**
  + Result: The proportion of **variance** explained by the PCA q-dimensional plane **equals to the sum of the first q eigenvalues of the covariance matrix**
* Implications:
  + PCA planes are nested: PCA 2D-plane contains PC1, PCA 3D-plan contains PCA 2-D plane
  + PC2 = second eigenvector of the covariance matrix
  + **PCs** are **linearly** **uncorrelated** 🡪 eigenvectors of a positive matrix are **orthogonal** to each other. More obs than variables (n >p) -> PCs orthonormal

**PCA in R**

pca\_res <- **prcomp**(data, center = TRUE, scale. = TRUE)

summary(pca\_res)

Plotting PCA in Regression:

* ratio of both rotations = slope of line: pca\_res$rotation
* need to define the line:
  + slope <- (pca\_res$rotation["y\_Name", "PC1"] \*-1) / (pca\_res$rotation["x\_Name", "PC1"] \*-1)
  + intercept <- pca\_res$center["yName"] – pca\_res$center["xName"] \*slope
  + geom\_abline(aes(slope=slope, intercept=intercept,color = "PC1"))

**Plotting PCA results in R**

* **Scree plot** 🡪 diagnostics tool whether PCA works well: shows variance in each projected direction
  + **plot**(pca\_res, type='l')
  + y-axis contains eigenvalues = amount of variation
  + 🡪 select which PCs to keep
* **Variant of scree plot:** plotting the proportion of the total variance for every principal component
  + **plot**(pca\_sum$importance[2,], type='l',
  + xlab='Principal components', ylab="Proportion of total variance")
  + **points**(pca\_sum$importance[2,])
* **Biplot**: projection of the data on **the first 2 PCs**
  + Includes position of each sample in terms of PC1 & PC2 and shows how initial variables map onto this
  + Correlation between variables = angle between vectors
    - **Small angle 🡪 high correlation**
    - F.e. all arrows negative for PC1: *All variables contribute with the same sign to the PC1🡪 PC1 correlates negatively with the projection of the axes of the variables in the original dataset (red vectors) 🡪 increase in variable implies a decrease in PC1 value*
  + biplot(pca\_res)
* access the projection of the original data on the principal components:
  + predict(pca\_res)

**PCA summary**

* statistical procedure that uses an orthogonal transformation to convert a set of possibly correlated variables into a set of linearly uncorrelated variables = PCs
* Each PC explains a fraction of the total variation in the dataset
* PC1 has the largest possible variance. Respectively, PC2 has the second-largest possible variance
* PCA aims to reduce the number of variables, while preserving as much information from the original dataset as possible. High-dimensional data is often visualized by plotting the first two principal components after performing PCA

Limitation of PCA:

* restricted to linear transformation of the data

Discussion:

* Clustering & dimension reduction belong to **unsupervised** learning methods
  + Supervised learning: regression, classification
* 🡪 discover patterns in data without being guided by some ground truths
* No “right” clustering/ subspace in real-life datasets
* Clustering and dimension reduction techniques are exploratory tools meant to help deriving some hypotheses
* These hypotheses are then best tested on independent data

Agglomerative = bottom-up

PC1 is orthogonal -> uncorrelated to PC2

# VI. Graphically supported hypotheses

**Descriptive plots** 🡪 exploring distribution of variables/data

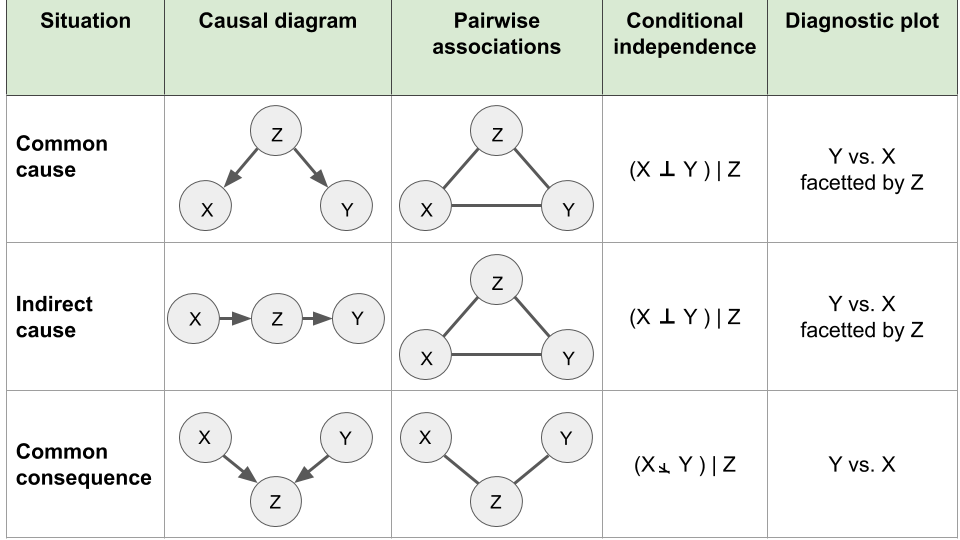
* One variable/univariate data: Histograms, single box plots or violin plots
* More variables/multivariate data: clustered heatmaps, PCA projections
  + Corresponds to Unsupervised learning: k-mean, HC, PCA
* **ggplot**(diamonds, **aes**(carat)) + **geom\_histogram**() + mytheme

🡪 can’t use those to make a claim!

**Associative plots** 🡪 relationships between variables, how they depend on each other

* y-axis = response variable; x-axis = explanatory variables
* Side-by-side boxplots, scatterplots
* Graphical representation of conditional distributions: p of y(response variable) **given** x (explanatory variable)
* **plot that graphically supports a hypothesis** and shows a causal relationship 🡪 show conditional distributions
  + Conditional distribution: for particular value (x) , I see a distribution on y-axis
  + When cond. distribution actually depends on x 🡪 x & y are dependent
  + F.e.: Scatterplot: trend that the response y increases in avg as x increases
  + F.e. Boxplot: show characteristic values of the distribution of y (quartiles, median) depends on the value if the category x
    - Corresponds to supervised learning: regression, classification
* **ggplot**(diamonds, **aes**(carat, price)) +
* **geom\_point**(alpha = 0.05) + *# alpha: point transparency*
* **stat\_smooth**() + mytheme

## **Correlation and Causation**

* One goal of data analysis: hypothesis about the **underlying causal mechanisms**
* Non-causal associations:
  + **The association is not statistically supported**
    - Observed association arose by chance
    - Reasons:
      * The association is driven by few data points (Graphically showing all data points including outliers can help; Hypothesis testing can be used to assess this possibility)
      * The dataset includes so many variables that the chance to have one pair of variables associating is high (called “data dredging”, “multiple testing”)
  + **The causal relationship is reverse**
    - **statistical dependencies are symmetric (if A correlates with B, then B correlates with A**) 🡪 mistake: claims where cause and effects are reversed. Examples:
      * People with healthier diet have higher blood pressure
      * Individuals in a low social status have a higher risk of schizophrenia.
      * The number of fire engines on a fire associates with higher damages.
      * Entering an intensive care unit increases your chances of dying.
    - All examples could be reversed…
    - No firm way to decide the direction of causality from a mere association 🡪 consider and discuss both possibilities when interpreting a correlation
  + **The association is induced by a third variable**
* 
* **Common Cause:** 
  + X independent of Y given Z
    - *## x <- z -> y*
    - **set.seed**(0)
    - n <- 1000 *# number of draws*
    - *# z: n draws of the binomial*
    - *# with 2 trials and prob=0.5*
    - z <- **rbinom**(n, 2, prob=0.5)
    - *# x: Gaussian with mean z*
    - **x** <- **rnorm**(n, mean=**z,** sd=1) 🡪 von z(**innen**) nach x (**außen**)
    - *# y: Gaussian with mean 2\*z*
    - y <- **rnorm**(n, mean=2\*z, sd=1). 🡪 von z nach y
    - *# gather into a data table*
    - *# we make z a factor for convenience with plotting functions*
    - dt <- **data.table**(
    - x,y,
    - z = **factor**(**paste0**("z=",z))
    - **table(z)**
    - x and y depend on z:
      * **ggplot**(dt, **aes**(x=z, y=x / y)) + **geom\_boxplot**()
    - **x and y correlate with each other due to common cause z:**
      * **ggplot**(dt, **aes**(x=x, y=y)) + **geom\_point**() + **geom\_smooth**()
      * **cor**(x,y)
    - **stratify by z 🡪 no association**
      * **ggplot**(dt, **aes**(x=x, y=y)) + **geom\_point**() + **facet\_wrap**(~z)
      * dt[, .(correlation=**cor**(x,y)), by=z] **🡪 computing cors**
    - 2 variables x and y are correlated, but their correlation is explained by z. Conditioned on z, their cor = 0.
    - 🡪 no Causal Relationship: Correlation never proves causation. Conditioning can rule out causation. (as in this case)
* **Indirect Cause:**
  + *# first step: tossing a coin once*
  + *# record x=1 if head, x=0 if tail.*
  + x <- **rbinom**(n, size=1, prob=0.5)
  + *# second step*
  + *# if x=0, toss the coin once, if x=1, toss the coins twice*
  + *# record z the number of heads of this second step.*
  + z <- **rbinom**(n, size=x+1, prob=0.5)
  + *## y: Gaussian with mean=z*
  + y <- **rnorm**(n, mean=z, sd=0.5)
  + dt <- **data.table**(
  + x = **factor**(**paste0**("x=",x)),
  + y,
  + z = **factor**(**paste0**("z=",z))
  + )
  + y associates with z
    - **ggplot**(dt, **aes**(x=x, y=y)) + **geom\_boxplot**()
  + **Association vanishes** whenwe **condition on z:**
    - **ggplot**(dt, **aes**(x=x, y=y)) + **geom\_boxplot**() + **facet\_wrap**(~z)

🡪 conditioning helped rule out causal relation between x & y

🡪 You **can’t differentiate between common cause and indirect cause generally from observational data**

* **Common Consequence** 
  + *#############*
  + *## common consequence x->z<-y*
  + *#############*
  + x <- **rnorm**(n)
  + y <- **rnorm**(n)
  + z <- x+y > 1
  + dt <- **data.table**(x,y,z)
* **Find if x and y correlate:** here it doesn’t
  + **ggplot**(dt, **aes**(x=x, y=y)) + **geom\_point**()
  + **cor**(x,y)
  + ***## x associates with y when z=TRUE***
  + **ggplot**(dt[z == TRUE], **aes**(x=x, y=y)) + **geom\_point**()
  + dt[z, **cor**(x,y)]

🡪 **Berkson’s** **paradox**: patients are in the hospital because they are sick, risk factors for their diseases, which do not correlate in the general population, happen to negatively correlate among them

🡪 **Association** **arises** once we **condition on the third variable**

* + conditioning should not be done systematically
  + data may already be conditioned we don’t know it

🡪 If you control for everything, include everything, you may induce correlations that were not there!!

## **Simpson's paradox. The sign (+ or -) designates positive or negative effects (causal diagram, left) or associations (right).Simpson’s paradox**

* variable X relates to Y in a certain way, but flips when stratifying for another var. Z

*# devtools::install\_github("easystats/correlation")*

**library**(correlation)

data <- **simulate\_simpson**(n = 100, groups = 4, r = 0.6) %>% as.data.table

**colnames**(data) <- **c**('X', 'Y', 'Z')

data

🡪 When visualizing the relationship between the variables X and Y, we can observe a negative correlation between X and Y.

**ggplot**(data, **aes**(X,Y)) + **geom\_point**() + **geom\_smooth**(method='lm') + mytheme

🡪 when **grouping by the variable Z** we observe a positive correlation **which is the opposite direction as before**

**ggplot**(data, **aes**(x = X, y = Y)) +

**geom\_point**(**aes**(color = Z)) +

**geom\_smooth**(**aes**(color = Z), method = "lm") +

**geom\_smooth**(method = "lm") + mytheme

## **Presentations in R**

**Color coding in R**

4 options for color coding with ggplot2:

1. Using default colors
2. Explicitly setting color names (e.g. “red”, “blue”)
3. Explicitly setting RGB or HTML color codes (e.g. 00-FF)
4. Explicitly setting color palettes

Use:

library(RColorBrewer)

display.brewer.all()

* **sequential palettes** for **continuous** variables to show quantitative differences
* **qualitative palettes** for **categorical** variables to separate items into distinct groups and we **use diverging palettes** for **numeric variables** that have a meaningful central value or breakout point (e.g. 0).

**General rules for color coding:**

Rule #1: consistent background

Rule #2: sufficient contrast for visibility

Rule #3: meaningful color usage

Rule #4: color usage with restraint

Rule #5: less is more

**Do’s in data visualization:**

* Keep visualizations simple
* Have meaningful and expressive titles
* Always label axes
* Keep the goal of the visualization in mind
* Know when to include 0

**Dont’s:**

* double encoding (color and axis encode the same)
* heavy or dark grid lines
* unnecessary text
* ornamented chart axes
* pictures within graphs
* shading or pseudo 3D plots
* Do not manipulate plots

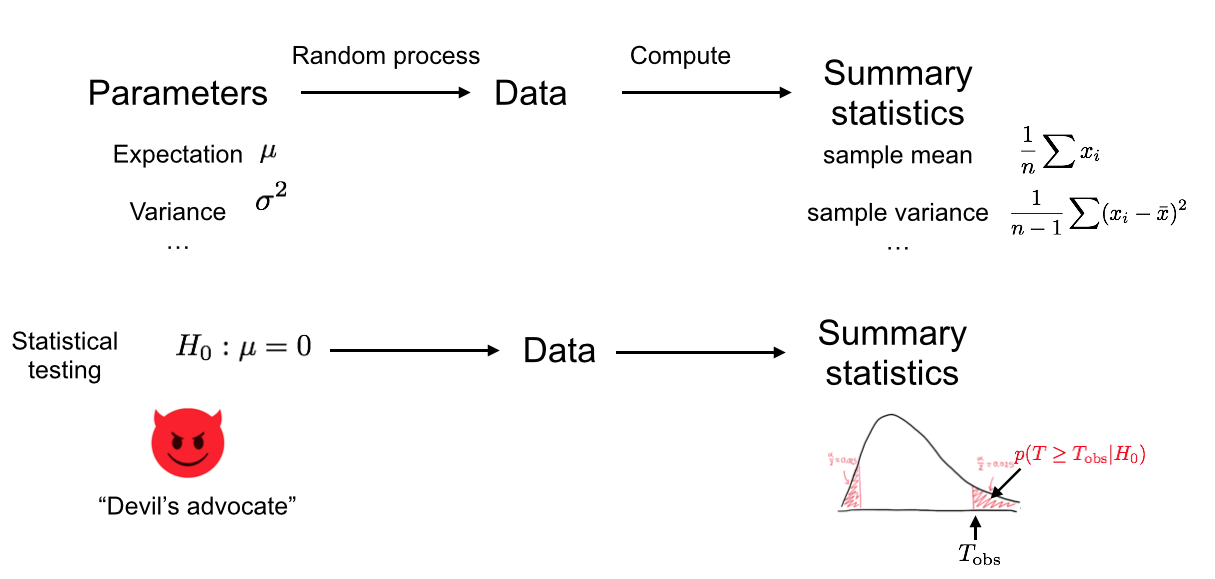
🡪 Think of **Data-ink ratio**: maximum of info with little as possible visual clues/ ink 🡪 more data and reduced visualizations

# VII. Resampling-based Statistical Assessment

What we found 🡪 is just by chance or statistically important

*Statistical significance*: trend is unlikely to have arisen purely by chance

## **Statistical hypothesis testing**

* 🡪 Did the observe trend arise by chance?
* 

**test statistic** = single number that summarizes the data and captures the trend. The more extreme the test statistic, the stronger the trend. (e.g. difference of medians or means)

* *Tobs / Tref*

**Null hypothesis** = a skeptical position/negative control: assuming observed trend is *not* real and arose purely by chance

**P-value =** probability of obtaining a test statistic the **same** as or **more extreme than the one we actually observe**d, under the assumption that the **null hypothesis is true**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* 2-sided tests: no clear direction of effects ahead known, such as ≠0
* 1-sided tests: only make sense if very good reason (before looking at the data!) that only the effect in one direction is important.
* The P-value is *not* the probability of the observed test statistic given that the null hypothesis is true
* P-value is *not* the probability that the null hypothesis is true given the data

The null hypothesis is said to be **rejected** for sufficiently **small P-values**

🡪 result is **statistically significant**

🡪 Common practice: significance level of α=0.05 and rejecting the null hypothesis if P<α. When the P-value is less than the chosen significance level, we **reject** the null hypothesis. But, in this framework, there is *no mechanism to accept* the null hypothesis. We can only **fail to reject** it.

* Presumption of innocence = Null Hypothesis
  + Only with strong evidence, we reject it
* One rejects it or one fails to rejects it (can have absence of evidence, this is not evidence of absence)

## **Permutation testing as Hypothesis testing**

* test the statistical **dependence** between two variables x and y (of any kind)
* test statistics can be any measure that captures the dependence.
* We assumed that the observations are **identically and independently distributed** (i.i.d) -> can be violated in practice, think of confounders leading to common cause, etc.
  + The data generating process is the same for all observations (identicallly distributed).
  + the observations are independent: the order of the indexing (the order of the rows of the data table) can be considered arbitrary
* ****The **null hypothesis** of a permutation test is that the two variables x and y are **statistically** **independent**
  + i.i.d Assumption: values of x could have occurred in any other order with the very same likelihood.
  + Large number of permutations 🡪 decent idea of the distribution
* For a one-sided p-value we do:
  + **m** be the number of random (Monte Carlo) permutations
  + **r** = #{T∗≥Tobs} be the **number of these random permutations that produce a test statistic greater than or equal to that calculated for the actual data**.
  + **Estimated 1-sided P-value:**
    - **Ein Bild, das Text, Uhr enthält.

      Automatisch generierte Beschreibung**If 0.01 = if we assume that the null hypothesis is true, the probability of observing a difference in median growth rates as, or more, extreme as the ones we actually observed, is less than one in one thousand. 🡪 need to be quite unlucky to get results like this by chance 🡪 reject the null hypothesis

dt\_permuted <- **copy**(dt)

**set.seed**(0) *# the seed of the random number generator*

dt\_permuted[ , genotype:**=sample**(genotype)]

*# The %+% operator updates the dataset of a ggplot object*

p <- p %+% dt\_permuted

diff\_median <- **function**(tab){

tab[genotype == 'Wild isolate', **median**(growth\_rate, na.rm=T)] -

tab[genotype == 'Lab strain', **median**(growth\_rate, na.rm=T)]

}

T\_obs <- **diff\_median**(dt)

T\_obs

**diff\_median**(dt\_permuted)

*# number of permutations*

m <- 1000

*# initialize T\_permuted with missing values*

*# (safer than with 0's)*

T\_permuted <- **rep**(NA, m)

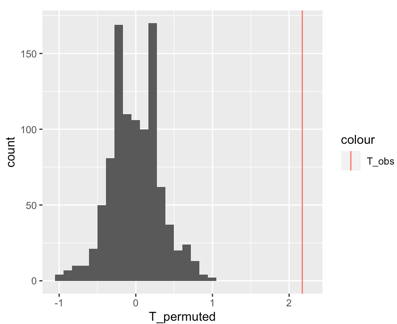
*# iterate for i=1 to m*

**for**(i **in** 1:m){

*# permute the genotype column in place*

dt\_permuted[ , genotype:**=sample**(genotype)]

*# store the difference of medians in the i-th entry of T\_permuted*

 T\_permuted[i] <- **diff\_median**(dt\_permuted)

}

**ggplot**( **data.table**(T\_permuted), **aes**(x = T\_permuted) ) +

**geom\_histogram**() +

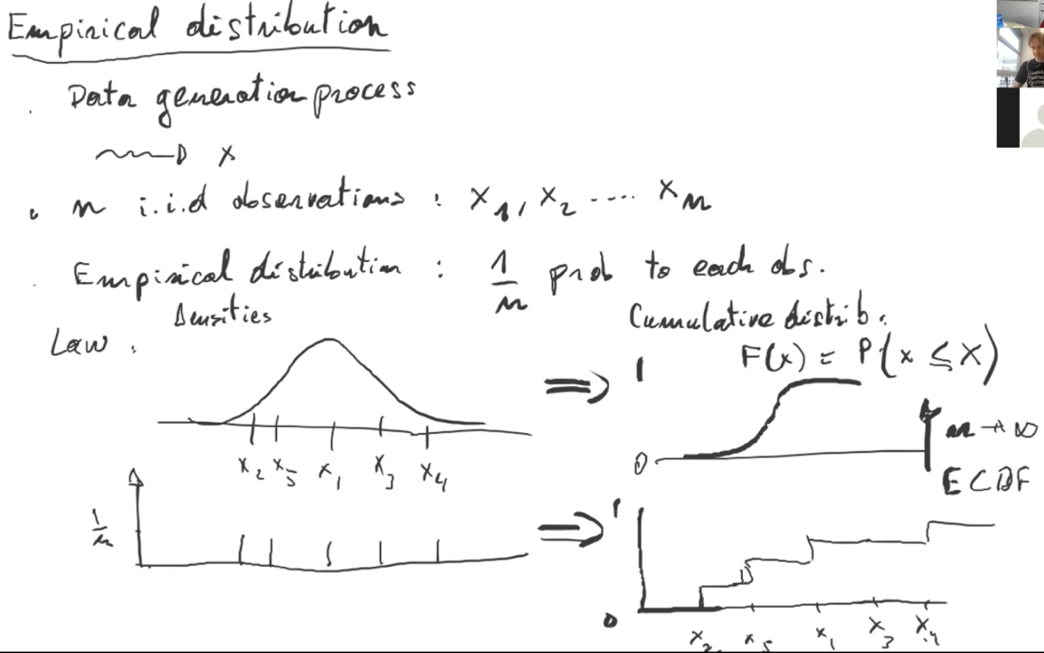
**geom\_vline**( **aes**(xintercept=T\_obs, color = "T\_obs") )

🡪 Permutation testing: the sampling procedure simulates the distribution of our test statistic under the null hypothesis

🡪*fix one column (smoker) and randomize symptoms:*

* *total number doesn’t change 🡪 raw total and column totals are fixed*
* *Hypergeometric distribution*

**Empirical Distribution**

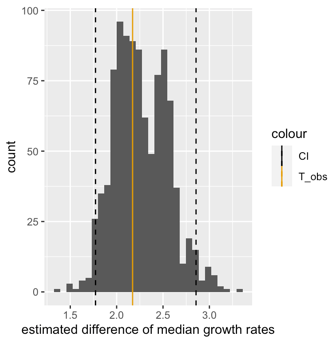
* Distribution, that gives equal probability of random variable x in random sample of n independent realizations: p = 1/n
* Draws = **sampling with replacement** 🡪 sample()
* x <- **rnorm**(20) *# 20 random numbers normally distributed*
* xrnd <- **sample**(x, 15, replace = TRUE) *# 15 random draws from data in x*
* 🡪 empirical distribution converges to the underlying distribution (function of x)
  + Empirical *cumulative* distribution (ECDF) = step function that jumps up by 1/n at each of the n data points
  + **plot**(**ecdf**(x))
* As n goes to infinity, the empirical distribution and the actual distribution will become more alike.
* 🡪 Drawing from the empirical distribution is a justified proxy for drawing from the actual underlying distribution. It is more accurate with large sample sizes.

## **Bootstrap Case Resampling for Confidence Intervals**

* **Based on drawing from empirical distribution function**
* Sample of size n **with replacement** from observed data to make a new dataset
  + Select some data points several times, and other points may not be selected at all
  + When n is not small, new data set will almost certainly not be the same
* *# number of random simulations*
* R <- 1000
* *# initialize T\_boot with missing values*
* *# (safer than with 0's)*
* T\_bootstrap <- **rep**(NA, 1000)
* *# iterate for i=1 to R*
* **for**(i **in** 1:R){
* *# sample the original data with same size with replacement*
* dt\_boot <- dt[**sample**(**nrow**(dt), replace=TRUE)]
* *# store the difference of medians in the i-th entry of T\_permuted*
* T\_bootstrap[i] <- **diff\_median**(dt\_boot)
* }

**95% bootstrap percentile confidence interval**

* conf\_int <- **quantile**(T\_bootstrap, **c**(0.025, 0.975))

**Plot** **entire distribution with the observed value and the 95% bootstrap percentile confidence interval**

**ggplot**(**data.table**(T\_bootstrap), **aes**(T\_bootstrap)) +

**geom\_histogram**(bins=30) +

**geom\_vline**(**aes**(xintercept=T\_obs, color = "T\_obs")) +

**geom\_vline**(**aes**(xintercept=conf\_int[1], color="CI"), linetype="dashed") +

**geom\_vline**(**aes**(xintercept=conf\_int[2], color="CI"), linetype="dashed") +

**scale\_color\_manual**(values=cbPalette) +

**xlab**("estimated difference of median growth rates")

## **Formal Confidence Intervals**

= method to **quantify our uncertainty about a parameter estimate**

= confidence level 1−α for a parameter θ is an **interval C=(a,b)**, which would the **data generation process be repeated**, **would contain the parameter with probability 1−α**, i.e. p(θ∈C)=1−α. Typical value is α=0.05 which leads to 95% confidence intervals

* If we repeat experiment 100 times: 95% conv = interval thus covers the estimates derived from 95 of our 100 experiments, and only excludes the 5 most extreme ones 🡪 plausible that the true median difference is in this interval, unless we got unlucky
* Compute a separate 95% interval for each of the experiments f.e. via bootstrap 🡪 true difference of medians, will be contained in about 95 of the computed intervals. In 5% of the time🡪 interval not to include the true difference of medians
  + 🡪 before we do an experiment, we have a 95% chance to end up with an interval that contains the true value

🡪 If our **null hypothesis** is that a given parameter, e.g. a mean**, is zero**, and our (1−α)% **confidence interval for this parameter does not include zero, we could say that we reject the null hypothesis at a significance level of α**. In that sense, hypothesis tests and confidence intervals are related.

**Formally doing Confidence Intervals in R:**

* based on standard normal distribution, true mean = 0:

**set.seed**(100)

exp\_1 <- **rnorm**(30) *# original data (30 values drawn from the standard normal distribution)*

*# Compute observed sample mean*

observed\_mean <- **mean**(exp\_1)

*# Do bootstrap and compute sample mean for each simulation*

boot <- **lapply**(1:1000, **function**(i){**sample**(exp\_1, 30, **replace** = TRUE)})

sample\_means <- **sapply**(boot, mean)

**95% case resampling boostrap confidence interval:**

*# 95% C.I. is given by the 2.5% and the 97.5% quantile*

conf\_int = **quantile**(sample\_means, **c**(0.025, 0.975))

*# Plot histogram*

bootstrap\_tbl = **data.table**(means = sample\_means)

**ggplot**(data = bootstrap\_tbl, **aes**(x = means)) +

**geom\_histogram**() +

**geom\_vline**(**aes**(xintercept=observed\_mean, color="observed")) +

**geom\_vline**(**aes**(xintercept=0, color="true")) +

**geom\_vline**(**aes**(xintercept=conf\_int[1], color="CI"), linetype="dashed") +

**geom\_vline**(**aes**(xintercept=conf\_int[2], color="CI"), linetype="dashed") +

**scale\_color\_manual**(name = "Legend", values = **c**(true="black", observed = "blue", CI = "red"))

Our example: **When 2 groups are compared**: we construct a 95% CI for median growth rate separately 🡪 reject Null Hypothesis if CIs do not overlap

It is important to note that this is *not* the same as rejecting if and only if the confidence interval for the difference of medians does not include zero, even if it may seem so intuitively. In fact, this “overlap” procedure is too conservative, and will fail to reject more often than the confidence level suggests. In the next chapter, we will give a technical reason for this.

Showing that the interval really captures true value about 95% of time when experiment repeated:

rerun\_experiment <- **function**(j) {

exp <- **rnorm**(30)

boot <- **lapply**(1:1000, **function**(i){**sample**(exp, 30, replace = TRUE)})

sample\_means <- **sapply**(boot, mean)

conf\_int = **quantile**(sample\_means, **c**(0.025, 0.975))

**return**(conf\_int)

}

rerun <- **sapply**(1:100, rerun\_experiment)

intervals <- **data.table**(**t**(rerun))

intervals[, idx :**=** 1:100]

intervals[, mid :**=** (`97.5%` + `2.5%`)/2]

intervals[, contains\_true :**=** ((`97.5%` >= 0) & (`2.5%` <= 0))]

**ggplot**(data = intervals, **aes**(mid, idx, color=contains\_true)) +

**geom\_errorbar**(**aes**(xmin=`2.5%`, xmax=`97.5%`)) +

**geom\_vline**(**aes**(xintercept=0)) + **xlab**("estimated mean") + **ylab**("experiment number")

# VIII. Analytical Statistical Assessment

**functions for distributions:**

* + random draws: rbinom, rnorm
  + density of probaliblity mass: dbinom, dnorm
  + cumulative distribution: pbinom, pnorm
  + quantile: qbinom, qnorm
  + uniform distribution: ppoints

Goal in data Analysis: Understand the underlying processes (biological, physical, economical,…) in order to predict and intervene

🡪 discover interesting **association/statistically depended between variables**

Reasons for misleading associations:

* **not robust**
* reverse causal direction
* common cause
* indirect effect
* common consequence

Permutation testing is limited in large data sets 🡪 **multiple testing**

* *large data sets 🡪 many (millions) of tests 🡪 we will falsely reject the null hypothesis a lot of times due to definition of p = 5% 🡪 5% will be rejected just by chance (8Mio tests 🡪 400k times) 🡪 have to go to lower levels*
* for 8 mio test, we do 8bn permutations 🡪 costly computing power and time

**3 Ingredients for statistical test:**

* **H0**, 1- or 2-sided
* Test statistic **T**
* Distribution of test statistic under H0 : p(T|H0)

## **Binomial Test: 1 binary variable hypotheses testing**

**Coin toss once:**

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**Head: X =1 of head; X = 0 when tail

🡪 2-sided p value:

🡪 Cannot conclude that null hypothesis is correct

**Tossing several times:**

* probability of one series is 0.5^n

**Binomial test in R**

test of a simple null hypothesis about the probability of success in a Bernoulli experiment

**binom.test**(tObs, numberOFTrails, p = 0.5, alternative = **c**("two.sided") )

* **Tobs** = number of successes
* Alternative: two.sided, greater or less
* Extracting p.value: test…$p.value

## Ein Bild, das Tisch enthält. Automatisch generierte Beschreibung**Fisher’s Test: 2 binary variables**

* Odds: a/b & c/d
* Ho: independence of the 2 variables
* Permutation testing: fixing one and permuting the other column
  + For each permutation: 2x2 contingency table:
  + Size of dataset, margins (= row and column totals) constant
  + **a**usually test statistic
    - distribution of a under H0 = **hypergeometric** distribution
      * Assumptions: **fixed margins**

Fisher test 🡪 exact instance of permutation testing

* Requires contingency table as input:

tbl = **data.table**(

severe = **c**(10, 10),

mild = **c**(20, 70)

)

tst <- **fisher.test**(tbl, alternative = "greater")

## **Binary-Continuous Variable Association testing**

### **Students t-test**

* **Ein Bild, das Text, Uhr, Messanzeige enthält.

  Automatisch generierte Beschreibungcompares**, (up to the constant c = sample size) the “**signal” of group difference** 🡪 estimated difference of the means of two groups, **to the “noise**”, i.e. how uncertain we are about our estimate of this difference.

x,y = sample means

s = pooled SD

c = constant, depending on sample size

* “noise” in our estimate is the ratio of the typical variations within the groups (s = pooled SD) over a term capturing the sample size (c).
  + 🡪 signal-to-noise ratio.
  + If t-statistic is large 🡪 clear difference in means.
  + if t-statistic is small 🡪 difference in means is not large compared to the noise
  + Larger sample size (more data) or larger between-group differences compared to within-group differences 🡪 larger t-statistics
  + Large variance 🡪 t-statistic gets small

**🡪 Student’s t-test:**

Assumptions:

1. All observations are **independent** of each other
2. X and Y (the two groups) follow **Gaussian** distributions
3. X & Y **same/equal** unknown **variance** (var.equal = T)



**H0: Equal means** of X and Y

Numerator in t-statistics is gaussian | SD estimate in denominator is noisy

🡪 the smaller the sample size, the noisier the estimate, the smaller degrees of freedom, the heavier the tails

**In R:**

**t.test**(growth\_rate ~ genotype, data=dt, alternative = …, **var.equal=TRUE** )

### **Welch’s test: Unequal variance**

**t.test**(growth\_rate ~ genotype, data=dt)

### **Wilcoxon rank-sum test**

Assumptions:

* All observations are **independent**
* Responses are ordinal, i.e. we can **rank** them
* **🡪 no longer Gaussian distribution assumed**

**H0**: the probability of an observation from the population X exceeding an observation from the second population Y equals the probability of an observation from Y exceeding an observation from X:

(two sided)

**wilcox.test**(growth\_rate ~ genotype, data=dt)

**Ranks are less sensitive to outliers than t-test (Gaussian Distribution Assumption)**

Data is indeed Gaussian 🡪 t-test more powerful

## **2 Continuous Variables**

### **Pearson correlation test**

* Captures **linear** relationship between 2 variables
  + between -1 and 1 (= all obs perfectly linearly related with a positive/negative slope)
* strongly influenced by outliers possibly

Assumptions:

* X & Y bivariate **Gaussian distribution**
* Observations of X & Y **i.i.d.**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**H0:** The two variables are statistically **independent**

**cor.test**(anscombe[,1], anscombe[,5], **method**="**pearson**")

### **Spearman rank correlation test**

* **without Gaussian distribution assumption** 🡪 can use rank correlation
* = Pearson correlation if **rank-transformed data**
  + **cor**(**rank**(anscombe[,4]), **rank**(anscombe[,8]), method="pearson")

Assumptions:

* no distributional assumptions

**H0:** The population rank-correlation is 0

**cor.test**(anscombe[,4], anscombe[,8], method="spearman")

🡪 Less powerful when data is actually Gaussian, but more robust to outliers and captures monotonic, yet non-linear relationships

## Ein Bild, das Text enthält. Automatisch generierte BeschreibungOverview of two-variable tests **Statistical Testing Summary**

## **Q-Q Plots: Assessing distributional assumptions**

* Checking whether data matches a distribution
  + Checking that data is normally distributed
* Histograms tricky: choosing right bin size
* Assuming data is **uniformly** distributed: 10% of the data is in interval [0, 0.1]
* Compare empirical quantiles with the values of the theoretical ones
  + **X-axis = expected quantiles, y-axis = observed quantiles**
* Computing deciles:
  + dec <- **quantile**(x, **seq**(0,1,0.1))
* When distribution matches the data, the points should be close to the diagonal
* If dots are on diagonal 🡪 can say it is uniformly distributed
  + Creating Observed quantiles: sort(rnorm(n)) == quantile (rnorm, probs =..)
  + Creating Expected quantiles: qnorm(ppoints(n))
  + **Order observed values(p\_val) for qq-plots**: order(p\_vals)
  + For **expected p\_values**: ppoints(p\_val)
    - Plot p-values in -log10 scale

Ranking data and getting quantile for every data point:

**ggplot**(

**data.table**(

x=**ppoints**(**length**(x)),

y=**sort**(x)

),

**aes**(x,y)

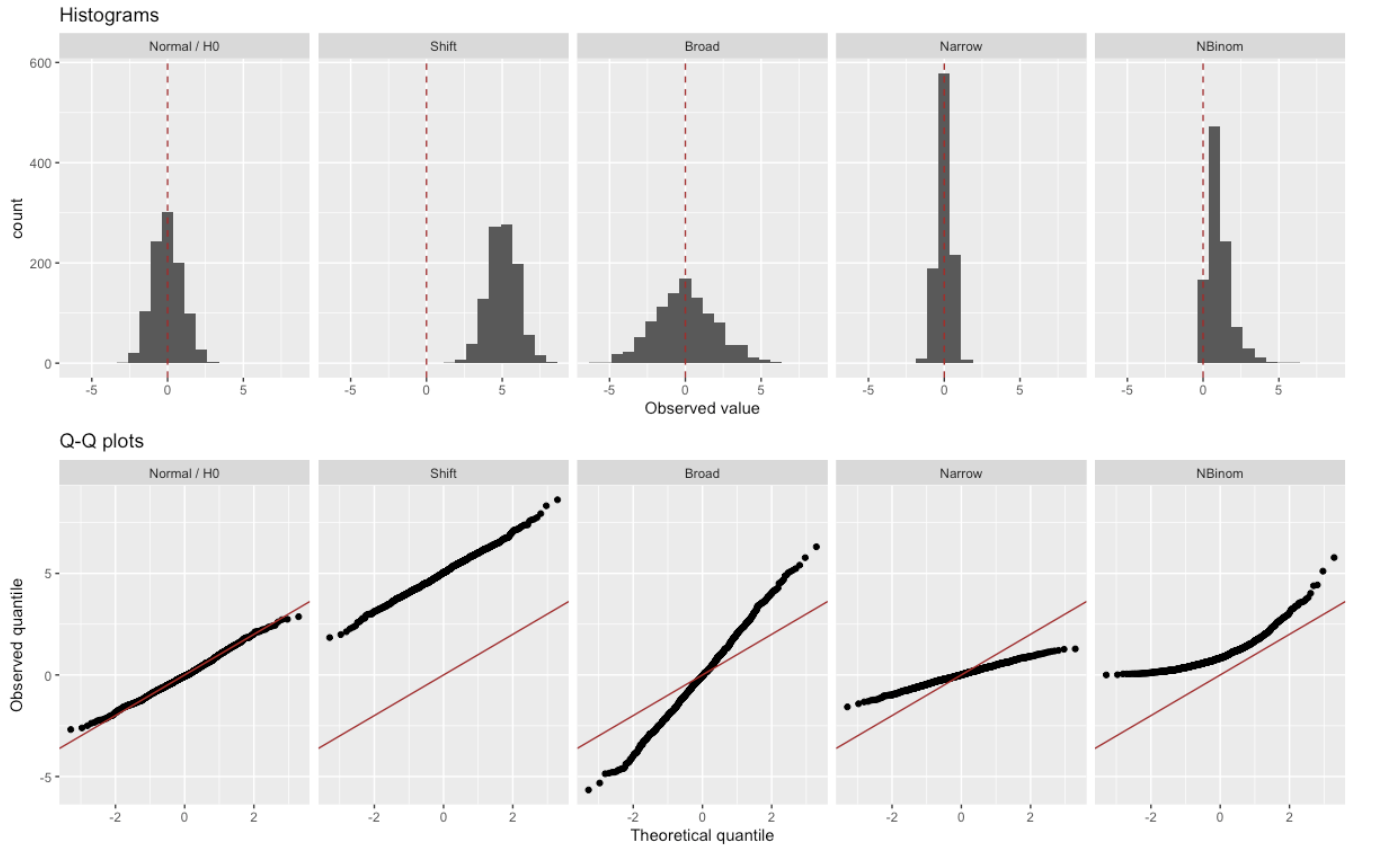
) + **geom\_point**() +

**xlim**(**c**(0,1)) + **ylim**(**c**(0,1)) +

**xlab**("Quantiles of the uniform distribution") +

**ylab**("Quantiles of the dataset") +

**geom\_abline**(intercept=0,slope=1) *## diagonal y=x*





Steeper QQ-Plot 🡪 SD high 🡪 Histo wider

Flatter QQ-Plot 🡪 SD lower 🡪 Histo narrow

# **IX. Statistical Assessments for Big Data**

Statistical significance != practical relevance

**Effect size**

* quantitative measure of the magnitude of a phenomenon (e.g. correlation, regression coefficient, mean difference,…)
* determines whether an effect is actually important
* 🡪 with bigger sample size 🡪 P-values drop
  + As n (samp. size) grows, any true effect, no matter how small, will be detected
  + Reality: many variables connected, effect size will rarely be 0.0
  + One can estimate how large a sample must be to significantly detect effects of certain amplitudes
  + With big data: one can get any small effect size statistically significant

🡪 Likely that whatever association we test, it will be significant. But many of these associations will likely be **too small** to be of **any practical importance**

🡪 **Problem of detecting very small effects**

Always Report: P-value, effect size and plot

**Multiple testing**

* if we test enough hypotheses at a given significance level, say α=0.05, we are bound to eventually get a significant result, even if the null hypothesis is always true
* Appears when stratifying large datasets into subgroups, testing a separate hypothesis for each one

**P-hacking/Fishing expedition**: testing hypotheses until one finally finds one that yields a significant P-value (=publishable result)

🡪 when we do so, apply a correction

Computing p-values for data.table:

dt[,

p.value :**=** **sapply**(

heads,

**function**(x)

**binom.test**(x, n = n, alternative = "greater")$p.value

)

]

Creating contingency table

**table**(dt[,.(fair, rejected = p.value<=0.05)])

## **Ein Bild, das Tisch enthält. Automatisch generierte BeschreibungBonferroni correction**

* Method to address multiple testing
* to control the **family-wise error rate (FWER):**
  + = p(V > 0); probability of at least one or more false positives (Type 1 Error) (prob. of 1 or more innocent killed)
  + f.e. FWER below a = 0.05
* **Under H0: if we do m tests and reject if P < a, then we will falsely reject m\*a times**
* To Control FWER: Bonferroni Correction: Only reject of P < a / m
* p.adjust(p\_values, method = “Bonferroni”)
* ensures we have few false positives, but comes at cost of many false negatives
* this control doesn’t require any assumption about dependence among P-values or about how many of the H0s are true
* Drawback: **very conservative**: 1000 tests (m) at a = 5%, we only reject if P < 0.00005
* Bonferroni produces less false positives but at the cost of considerably more false negatives

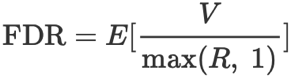
In R data.table:

dt[ , bonferroni :**=** **p.adjust**(p.value, method="**bonferroni**")]

new contingency table

**table**(dt[,.(fair, rejected = Bonferroni <= 0.05)])

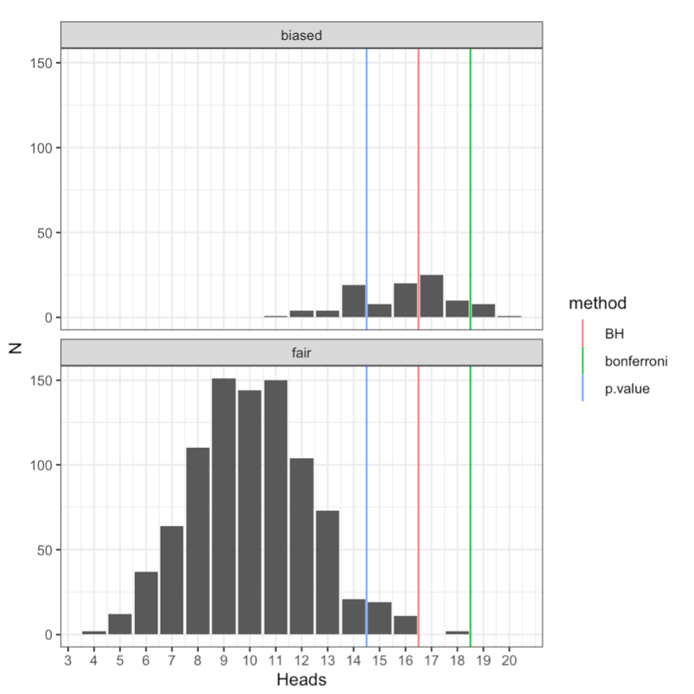
## **Benjamin-Hochberg correction**

* Controls **False discovery rate (FDR)**
  + = expected fraction of false positives among all discoveries
* p.adjust(p\_values, method = “BH”)
* **Less conservative** than Bonferroni

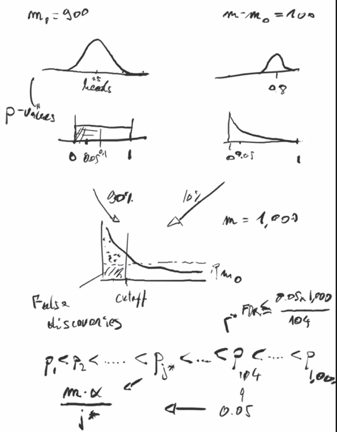
In R data.table:

dt[ , BH :**=** **p.adjust**(p.value, method="**BH**")]

**🡪 When H0 is true: P-values uniformly distributed (assuming independence)**



* FWER / Bonferroni is most stringent
* FDR / Benjamin-Hochberg is intermediate



## Creating a matrix of p-values for every combination:

* pv <- **apply**(onWhichMatrixTested, 2 (for columns, 1 for rows) , **function**(x){

**rowttests**(**as.matrix**(otherVariableMatrix), fac = **factor**(x))**$**p.value } )

🡪 Adjusting it for **FDR**:

* fdr <- **matrix**(**p.adjust**(**as.vector**(pv), method=”BH”), ncol = **ncol**(pv))
* *## get the locations of significant points*
* signif = **which**(fdr **<** 0.1, arr.ind = TRUE)
* *## eQTL mapping plot*
* signif <- **as.data.table**(signif)  
  **colnames**(signif) <- **c**(’gene\_index’, ’marker\_index’)

# X. Linear Regression

Does y (response variable) depend on xj (explanatory) everything else being the same?



**🡪 Probability of y given x**

Generally, the H0 above is impractical because:

* Need to be able to condition on continuous variables
* Need to deal with any type distribution
* Combinatorial explosion of strata: for every new factor, more strata, loss of statistical power

**Linear regression assumptions:**

1. **Errors e** (& Residuals ê) **are identically and independently distributed**
2. **Errors** e follow a **normal distribution** ( = Conditional distribution is **Gaussian** distribution
   1. The Gaussian assumption of the **errors** is key to all statistical tests. An implication that the errors follow a Gaussian distribution is that the residuals also follow a Gaussian distribution.
3. Conditional expectation of y is a **linear combination** of the explanatory variables

* 
  + Assumptions allow us to deal with continuous & discrete explanatory variables
  + Effects of explanatory variables do not depend on the value of other variables; Effects do not change between strata

***Error*** *= difference between the observed value and the true value (unobserved)*

***Residual*** *= difference between the predicted value (by the model) and the observed/ actual value*

*🡪 only have access to residuals*

🡪no categorical explosion of strata

**Limitations of assumptions:**

* Gaussian assumption limiting: cannot deal with binary response variables
* Assumption that variance is independent of strata can be problem

**Applications** of LR:

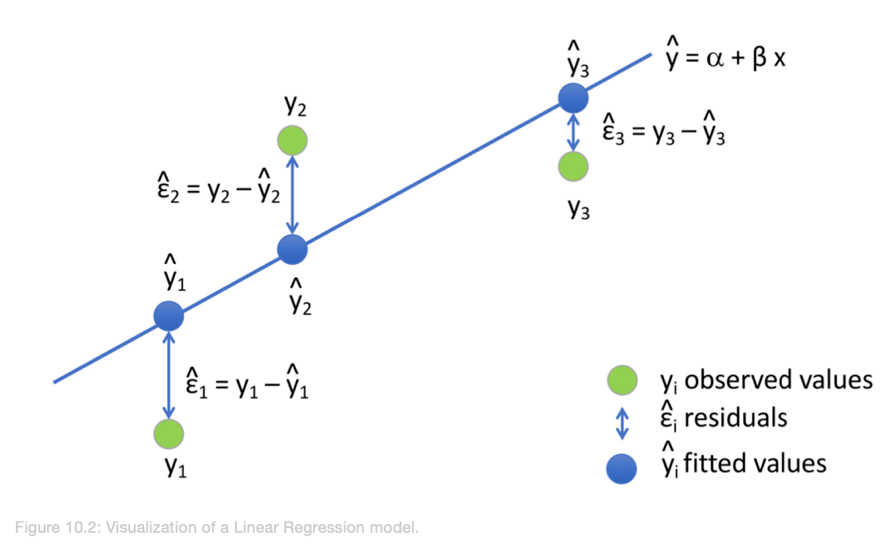
* Test conditional dependence: H0:
* Estimate effects of one variable on the response variable: Estimating beta
* Predict the value of y
* Quantify how much variation of y can be explained by explanatory variables

## **Univariate Regression**

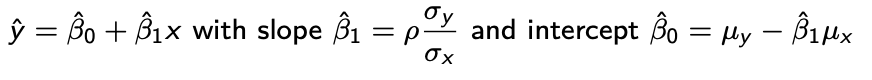
* Probability of y given x 🡪 Stratifying

**Fitting** a Linear Regression, estimating the parameters, is based on **Maximum Likelihood Principle:**

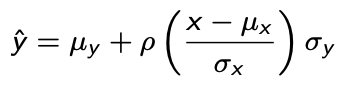
* Choose parameter values for which the data is most probable
* Assuming conditional independence, likelihood (probability of the data given the parameter)
* Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungWe assume Gaussian: Max likelihood of this model (with Gaussian assumption) = **Minimizing the sums of the squared errors** (Least squares estimates**) = RSS**
  + Least squares criterion as model fitting criterion
  + **Error** = difference between observed value and expected (true) value
  + Ein Bild, das Text enthält.

    Automatisch generierte Beschreibung



**Result:**



**Interpretation of fitted coefficients:**

* (Pearson) Correlation between x&y = 1: regression line predicts an increase in y by the same number of standard deviations
* Cor = 0: Don’t use x for prediction; predict only population average
* Cor between 0 - 1: prediction of y is closer to its average than x to its average (in standard units)
* Cor = -1; predict reduction instead of increase

**In R:**

* Lm\_model <- lm(y ~ x, data = …)
* coefficients(Lm\_model)
* in ggplot: + geom\_smooth(method = “lm”)
* Prediction: lm-object as input, returns prediction:
  + y\_hat <- predict(Lm\_model, se.fit = TRUE)
  + names(y\_hat)
* summary(lm\_model)$coef
* Feature Matrix: model.matrix(Sepal.Width ~Sepal.Length \* Species, data=iris\_dt)
* *Regression towards the mean* in extreme scenarios: children of extremely tall parents still tall relative to general population, but smaller than parents usually -> regress towards mean

How well does the model represent our data:

* R^2: *coefficient of determination* = percentage of variance explained by the model
  + R^2 = 1 – (Residual sum of squares / sum of squares)
  + 1: Residuals are zero, Model fits perfectly
  + 0: Model learned no variation, but at least the global mean: RSS = SS
* In R: summary(lm\_model)$r.squared

## **Multivariate linear regression:**

* = Linear regression against multiple variables
  + E.g. stratifying by one effect to see if subgroups still show same slope (beta)
  + E[*R* | *BB* = *x*1, *HR* = *x*2] = *b*0 + *b1x*1 + 2*bx*2
* Explanatory variables are assumed to be independent
* In principle same as for univariate
* Parameters (betas) estimated using **LSE**:
  + LSEs are **unbiased:** if the data truly originates from such a data generative model, the expected value of the estimates over repeated random realizations equals the true underlying parameter values
  + Estimates are **consistent**: Converge to the true values with increasing sample size
  + True, even if explanatory variables are correlated (unless perfectly, then parameters become not identifiable)

**In R:**

* lm\_model <- lm(y ~ x **+ x**, data = …)
* Coef(lm\_model)
* Summary(lm\_model)
* beta\_hat <- replicate(1000, sim(alpha,beta,sigma\_sq,x)$coef["x"])
* Computing the fitted values or predict response for new data: Predict(lm\_model)
* Compute the residuals: resid(lm\_model)
* slope = get\_slope(y,x)
* Stratifying:
  + dat <- Teams\_filt[,HR\_strata :**= round**(HR/G, 1)][HR\_strata >= 0.4 & HR\_strata <=1.2]
  + Getting slopes:
  + dat[**order**(HR\_strata),.(slope = **get\_slope**(BB\_per\_game, R\_per\_game)), by='HR\_strata']

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## **Nested Models:**

* = Testing an entire set of variables:
* Comparing a full model to a reduced model where the reduced model is a special case of the full model
* a model A is nested into a model B if the parameters in A are a subset of the parameters of model B
  + In Multivariate regression: setting a set of **parameters to 0**
* *y* = *b*0 + *b*1*x*1 + *bx*2 + *bx*3  
  - For testing whether the coefficients of two explanatory variables *x*1 and *x*2 should be 0:
  + Full model: all *b*’s can take any value.
  + Reduced model: *b*1 = *b*2 = 0 (only the mean *b*0 and the third parameter *b*3 can take any value).

### **F-Test/ANOVA**

* comparing 2 nested linear regressions = comparing the fit improvements as measured by the change residual sum of squares
* the larger model, which has more parameters, always fits better to the data
* ask whether this improvement is significant under the null hypothesis, that the smaller model is the correct one
* F-statistic distributed according to F-distribution
* More parameters 🡪 Least Sum of Squares is reduced

Ein Bild, das Text enthält.

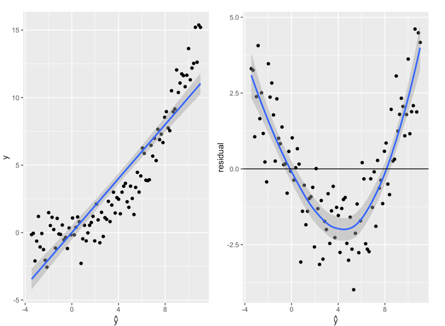
Automatisch generierte BeschreibungExample:

* We want to test the effect of both indicators at the same time:
* ****
  + Full model: is the space where all three b can take any value.
  + Reduced model: is the space where only b0 can take any value.
* as.factor(categories)
* full\_model <- lm(y ~ x, data = …)
* reduced\_model <- lm(y ~ 1, data = …) 🡪 only intercept 1 in this case
* anova(full\_model, reduced\_model)
* Interpretation: F-value in full model > significance level 🡪 full model models data significantly better than reduced model. There is a difference in the means of the (here 3) groups

## **Diagnostic Plots**

### **Residual plots: non-linearity**

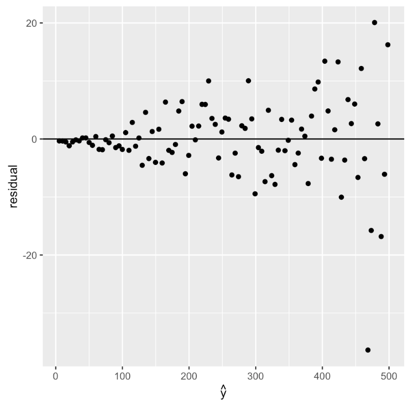
1. Assessing non-linearity: noticed as the average of the residual depends on the predicted values (plots the **residual against** *(y-axis*) the **predicted response values** *(x-axis*)
   * geom\_smooth(method = lm) helps spotting it
   * resid(lm\_model)
   * predict(lm\_model)

**Implications of non-linearity:**

* Predictions are suboptimal
* Explained variance is underestimated
* i.i.d assumption is violated
* Conditional dependencies can be affected

**If non-linearity is revealed:**

* Transform explanatory variables or the responses (difficult)
* In R:
  + model <- lm(y ~ poly(x,3)) 🡪 fits a polynomial degree of 3



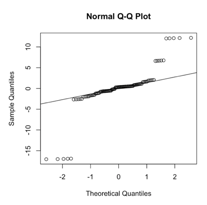
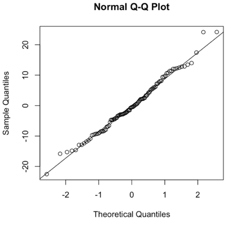
### **Heteroscedascity: error variance is not constant**

* variance of residuals not constant across all data points
* **violates the i.i.d assumption of the errors**
* **Residual plots** help spotting: plots the residual against the predicted response values
  + x <- 1:100  
    y <- **rnorm**(100, mean=5 \* x, sd=0.1\*x)  
    m <- **lm**(y ~ x)  
    **ggplot**(data=NULL, **aes**(**predict**(m), **resid**(m))) +
  + **geom\_point**() + **geom\_abline**(intercept=0, slope=0) + **labs**(x=**expression**(**hat**(y)), y="residual")
* what to do: transform response variables y (difficult), weighted-least squares, generalized linear models

### **Q-Q-Plots of the residuals: Gaussianity**

* Gaussian assumption of errors is key to all statistical tests 🡪 residuals also follow Gaussian distribution
* QQ-Plot of the residuals against the normal distribution
* In R:
  + **set.seed**(0)  
    x <- 1:100  
    y <- **rnorm**(100, mean=5 \* x, sd=10)
  + m <- **lm**(y ~ x)
  + **qqnorm(residuals(m))**
  + **qqline(residuals(m))**
  + ggplot(prediction\_dt, aes(sample = residuals)) + geom\_qq() + geom\_qq\_line()

Good: Bad:

****

🡪 If Gaussianity of residuals not satisfied:

* With enough data, the regression lines might not be too severely affected the least squares estimates converge to the expected values. Applying least squares to fit and predict does not depend on the Gaussian assumption!
* Hypothesis testing can be flawed

🡪 What to do:

* work with fundamentally non-Gaussian data (like Poisson distributed data, which are count data), or data with long tails and outliers, generalized linear model with another distribution, case resampling to estimate confidence intervals

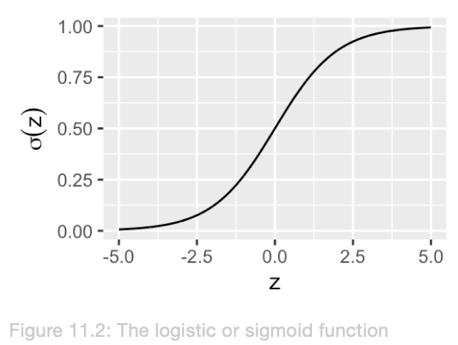
# XI. Logistic Regression

Linear regression is not adapted to categorial responses 🡪 **response y is a now category**

* E.g. having disease or not, spam email or not, etc
* No solution for classification with linear regression: We have Infinite bins
* prediction tasks for which the response is a category are called ***classification***tasks
  + will focus on **binary classification (exactly 2 categories): y {0,1]**

**Ein Bild, das Text enthält.

Automatisch generierte BeschreibungClassification task** f.e. predict the student’s sex given the student’s height.

* **Odds** for binary variable y defined as:
  + Ratio of (probability of class 1) / (probability of class 0)
* Plotting **odds** in **logarithmic** **scale** suggests linear relationship
* Intercept moves up by log odds ratio
* Slope captures separation
* Function for classification tasks = *logit*. Its **reverse** is ***logistic* or *sigmoid* function**
  + Sigmoid function has S shape, is symmetric and maps real numbers to the [0,1] interval
  + 🡪 linear relationship!

In **Linear Regression**: the expectation of the outcome conditioned on the features is obtained as a linear combination of the features

In **Logistic Regression**: predicting the probability of a class given some features

* Using the logistics function, which maps linear combinations of features to the [0,1] interval
* Probability **μi** is the **expectation of success class (y=1)** **conditioned** on the **features**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

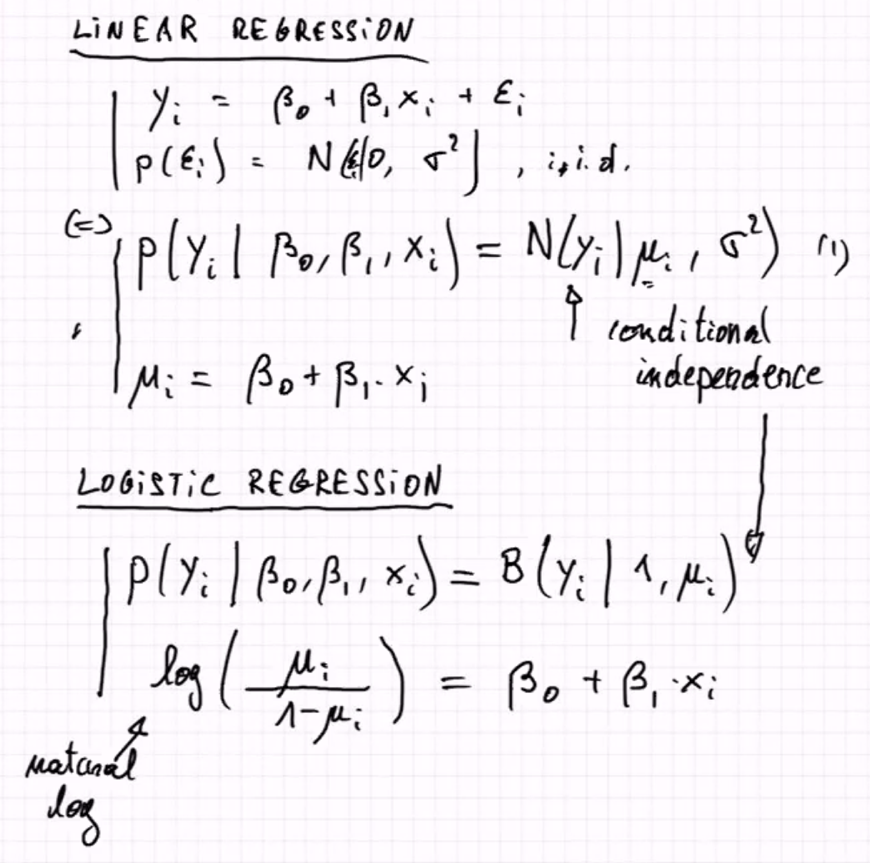
B= binomial distribution: 1 trial, probability μi = Bernoulli distribution:

* discrete distribution with 2 outcomes: success (y=1), failure (y=0)

**Cross-entropy** between model predictions (predicted prop μi) and observations y:



* In Classification, **we minimize cross-entropy**, not squared errors



Both formulas are equivalent: Second

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## **Logistic regression as a generalized linear model:(GLM)**

* changed linear regression by applying a transformation to the linear combination of the features in order to use it for another type of predictions (classification)
* Logistic regression is one instance of generalized linear models:

1. A probability distribution from the exponential family
   * + Logistic regression: **Bernoulli**
2. A linear predictor n = Xb
   * + **Logistic regression: logit(μi) = ni = b0 + Sums(bjxij)**
3. A link function g such as E(y) = μ = g^-1(n)
   * + Logistic regression: **g = logit** and g^-1 = sigmoid

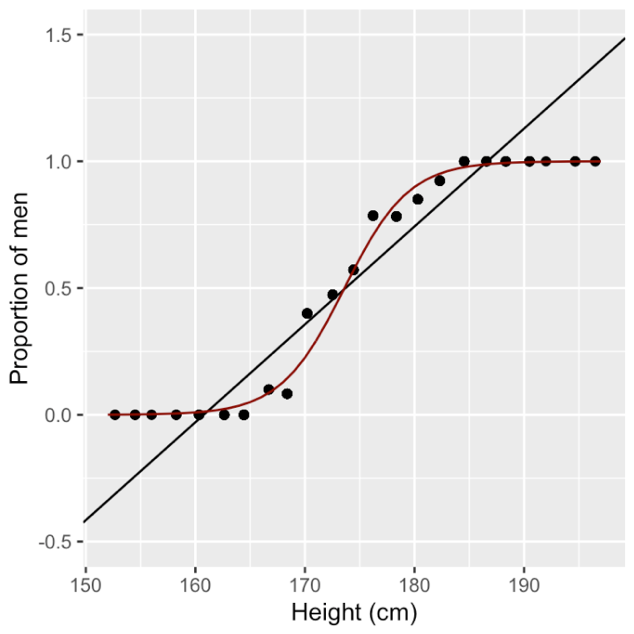
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* inverse of the link function (activation function):in logistic regression is the logistic function.

**In R:**

* logistic\_fit <- **glm**(y ~x, data, family = “binomial”)
* apply model to data with predict():
  + heights[, µ\_hat := **predict**(logistic\_fit, heights, type="response")]

predicted values using logistic regression (red) compared to a linear regression of the proportions (black). 🡪 logistic regression fits better to our data and fixes the issue of having predictions outside the [0,1] interval.

**Predicted odds:**

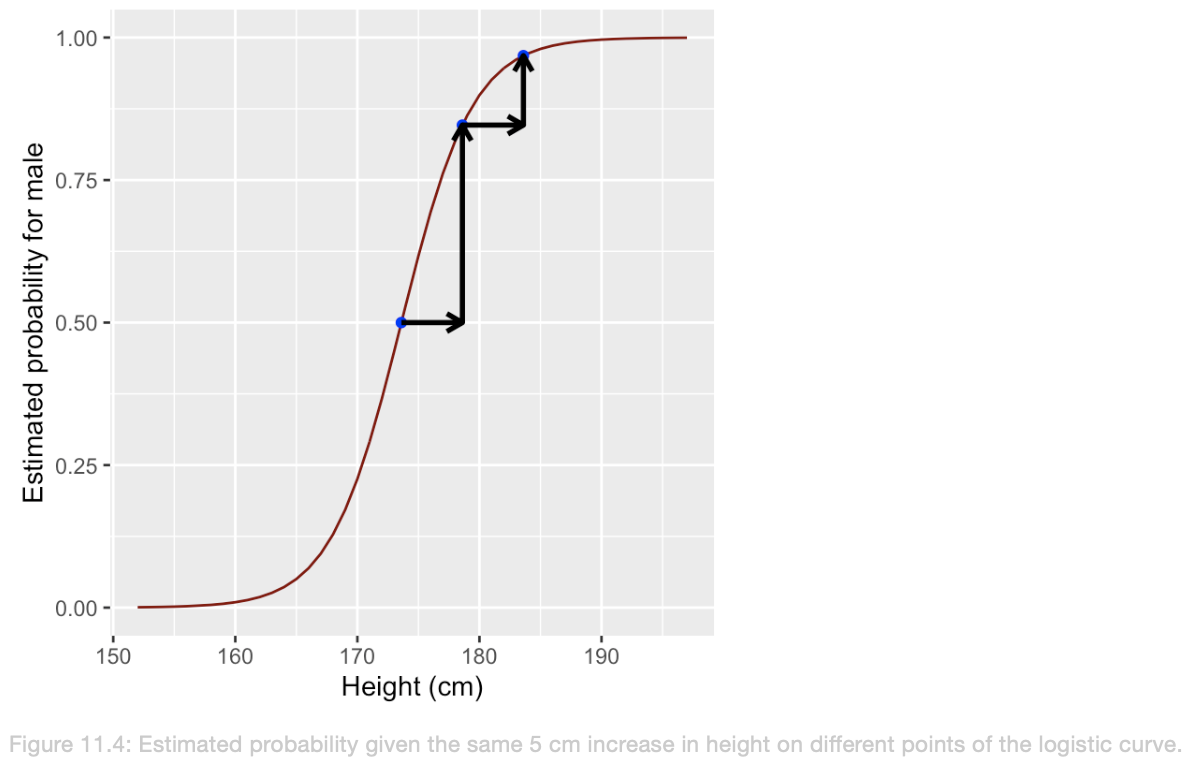
* logit of predicted response for a certain input is predicted log odds for the positive class (y=1) for that input
* **log**\_odds\_178 <- **predict**(logistic\_fit,data.table(height=178))
* to get **odds**: exp(log\_odds\_178)

### **Coefficients - Interpretation**

* b values in logistic regression = **log odds ratios** associated with an increase by one unit of the corresponding explanatory variable
  + 🡪 **Odds** ratios: apply exp().
* Coef(logistic\_fit)
* Ein Bild, das Text enthält.

  Automatisch generierte BeschreibungExp(coef(logistic\_fit)[2]) 🡪 Odds ratio
* odds ratios: by what ratio the odds of the outcome change when changing the predictive variable by one unit.
* Scale each feature by sd to compare feature odd ratios
* in logistic regression the log odds is predicted as a linear combination of the features, where the **coefficients are log odds ratios**.
  + **increasing the height by h centimeters changes the log odds by h x 0.342**, = it **multiplies the odds by e^hx0.342 = 1.408^h.**

Logistic R: Relationship between input & predicted value is **not linear**!



**Class imbalance**

* one class havening more instances than the other
* intercept of the logistic regression is now lower and lines are nearly parallel
* **log odds** in imbalanced logistic for **underrepresented** group generally **lower**
  + odds male:female per stratum proportionally change when the overall population odds change
  + 🡪 changes in class imbalance affects the intercept b0 of a logistic regression by adding a constant

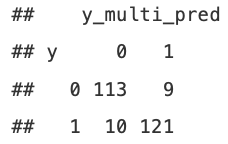
**Multiple Logistic regression:**

multi\_logistic\_fit <- **glm**(y ~ height + mother + father, data=heights, family = "binomial")

* Ein Bild, das Text enthält.

  Automatisch generierte Beschreibungan increase in student’ts height increases the odds for male
* negative coefficients to the height of the mother and of the father; the taller either parent, the taller the children. more likely that a tall person is a female, if the parents are tall.

**Classification with logistic regression**

* logistic regression predicts probability
* Classification = attribute one **class** per instance, not probability
* Hard classification: if **μ > 0.5, predict class 1,** else predict class 0
* round() 🡪 to get binary variable, implicit threshold 0.5
  + heights[, y\_multi\_pred :**= round**(**predict**(multi\_logistic\_fit, heights, type="response"))]
  + heights[, **table**(y, y\_multi\_pred)] **🡪 confusion matrix**
  + 
  + 10 **males** predicted as females (false negatives), 9 females predicted as males (false positives)
* data\_dt[, pred := predict (logistic\_fit)]
* Getting probabilities in range [0,1]:
  + data\_dt[, pred := predict (logistic\_fit, type = "**response**")]
* Getting Binary variable:
  + data\_dt[, pred := **round**(predict (logistic\_fit, type = "response"))]

## **Confusion Matrix**

**False Negative:** **Predicted** negative, but actually positive

In R:

data\_dt[,**table**(Outcome, **prediction\_logreg > threshold**)]

* as function:
* confusion\_matrix <- function(dt, outcome\_column, prediction\_column, threshold){ return(dt[, table(get(outcome\_column), get(prediction\_column)>threshold) ])}
* use *get()* to get the column from a string
* on multiple thresholds simultaneously:
  + lapply(thresholds, **function**(t){confusion**\_**matrix(diabetes\_dt, "preds\_model1", "Outcome", t)

Performance Metrics

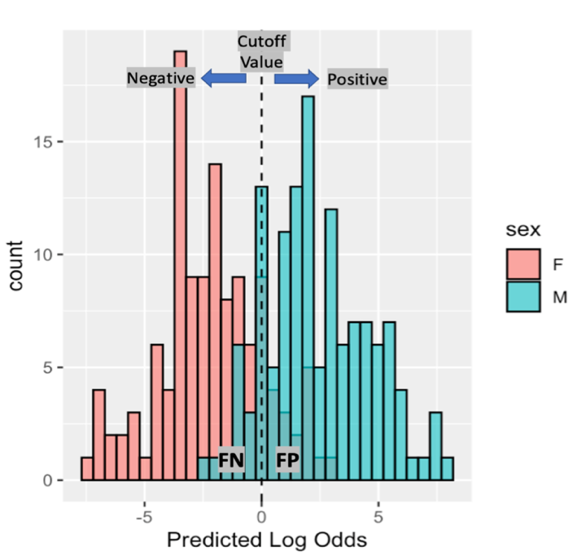
**Sensitivity** *(****recall****,* ***true positive rate***) = fraction of actual positives that is predicted to be positive

* TP / P = TP / (TP + FN)

**Specificity** (*true negative rate*) = fraction of actual negatives that is predicted to be negative

* TN / N = TN / (TN + FP)

**Precision** (*positive predictive value*)= fraction of predicted positives that are indeed positives

* = TP / (TP + FP)

*FDR = 1 – E[Precision]*

**Cutoff Value**

* Separates to some extent positive from negative class
* Choice of cutoff influences performance metrics
* Tradeoff between sensitivity & specificity

## **ROC curve**

* = receiver operating characteristic
* = evaluating the quality of a binary classifier at different cutoffs
* Ein Bild, das Text enthält.

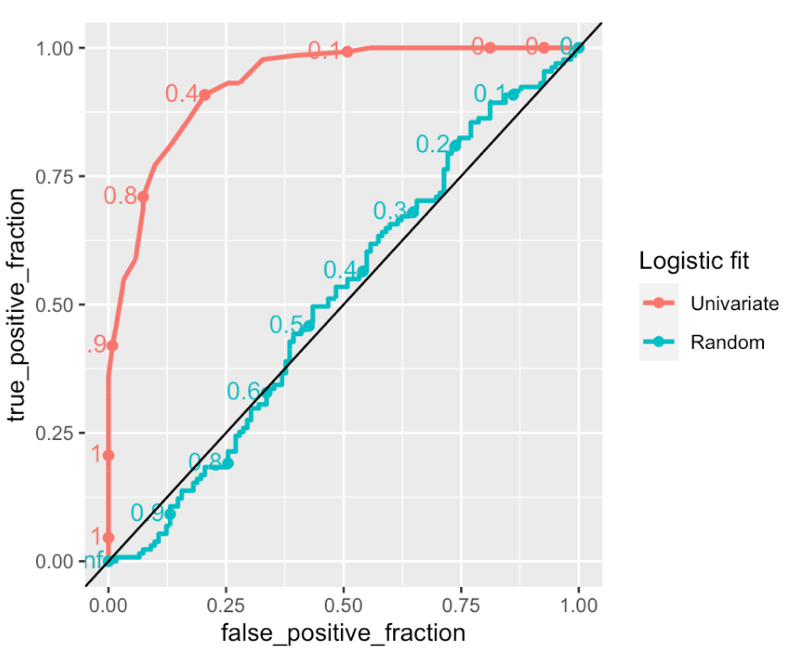
  Automatisch generierte Beschreibung**X**-axis: **False positive rate** (1-specificity)
* **Ein Bild, das Text enthält.

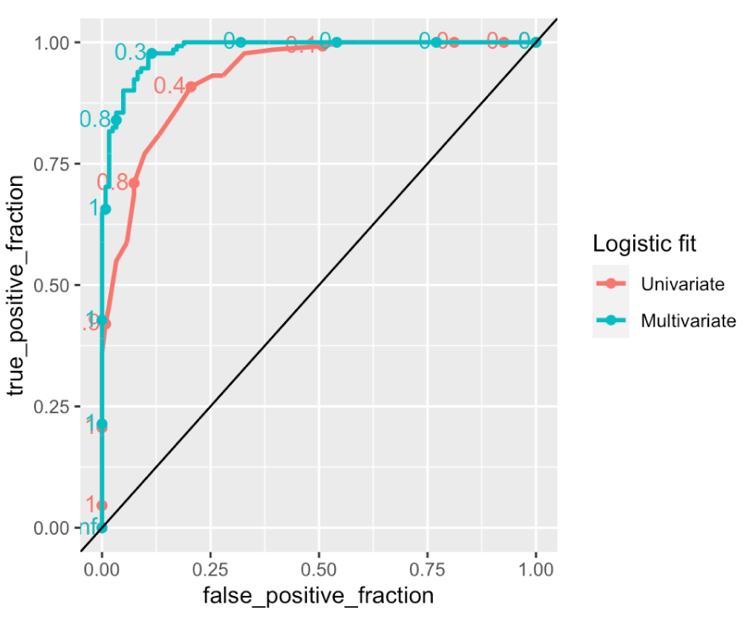
  Automatisch generierte BeschreibungY**-axis: **True positive rate** (sensitivity)
* Outcome must be **as.numeric()**

**library**(plotROC)  
heights[, random\_scores:**=runif**(.N)]

**dt\_melted** <- heights[, .(y, mu\_hat, random\_scores)] %>% **melt**(id.vars="y", variable.name = "logistic\_fit", value.name="prediction")

**ggroc** <- **ggplot**(dt\_melted, **aes**(d = as.numeric(Outcome), m = prediction, color=logistic\_fit)) + **geom\_roc**() + **geom\_abline() + scale\_color\_discrete**(name = "Logistic fit", labels = **c**("Univariate", "Random"))

* points = cutoff value
* Cutoff = 0: all instances classified as positive 🡪 FPR & TPR = 1
* Cutoff = 1: all instances classified as negative 🡪 FPR & TPR = 0
* Random classifier: diagonal

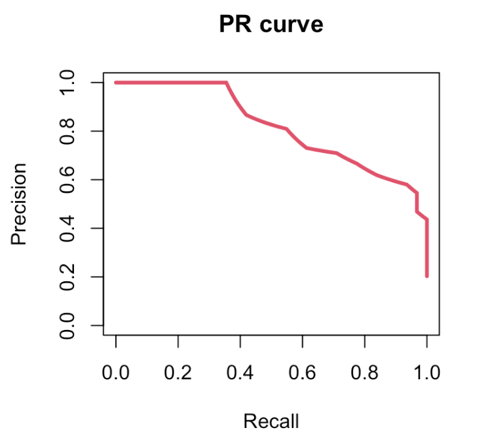
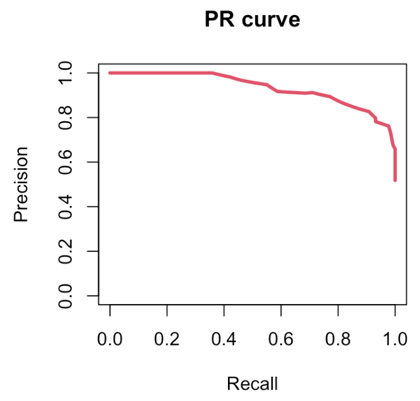


🡪 for lower FPR: TPR higher in **multiple** logistic regression model 🡪 better performance

AUC = Area under the ROC curve

* Classification performance metric
* 1 = model perfectly able to distinguish between positive and negative classes
* 0.5 = AUC of diagonal ROC = classifier no better than random classification
* aucs <- as.data.table(calc\_auc(ggroc))

**Precision Recall Curve**

* plots precision (y-axis) against recall (x-axis)
* package PRROC
  + **library**(PRROC)
  + PRROC\_obj <- **pr.curve**(scores.class0 = heights$mu\_hat, weights.class0=heights$y, curve=TRUE)
  + **plot**(PRROC\_obj, auc.main=FALSE , color=2)

Imbalanced dataset Balanced dataset

🡪 imbalanced for the female (negative class), the logistic regression model **provides lower precisions for the same recall** = such model is worse at classifying males.

* The PR curves used in very **imbalanced situations** in which the positive class is strongly underrepresented.
* for “Finding a needle in a haystack” situations, precision-recall curves emphasizes better the performance of the model among the top scoring predictions than the ROC curve.

# **XII. Supervised Learning**

* **interested in good predictions** rather than identifying the most predictive features or drawing conclusion about stat. independence
* **supervised learning** = goal is to build a powerful algorithm that takes feature values as input and returns a prediction for an outcome, even when we do not know the value for the actual outcome.
  + train an algorithm using a data set for which we know the outcome, and then use this trained model to make predictions.
  + When building the model, we associate each set of feature values to a certain outcome
  + fitting conditional distribution p(y|x) where y is the outcome and x are the features 🡪 Linear & Logistic Regression
* **Unsupervised learning** = we do not associate feature values to an outcome
  + we lack an outcome variable that can supervise our analysis and model building
  + unsupervised learning algorithms identify **patterns** in the distribution of data
  + clustering problems and dimensionality reduction are attributed to unsupervised machine learning problems
  + fitting the distribution of the data p(x) (covariance: PCA, mixture components: clustering, etc.)

Approach:

y = outcome (response) we want to predict

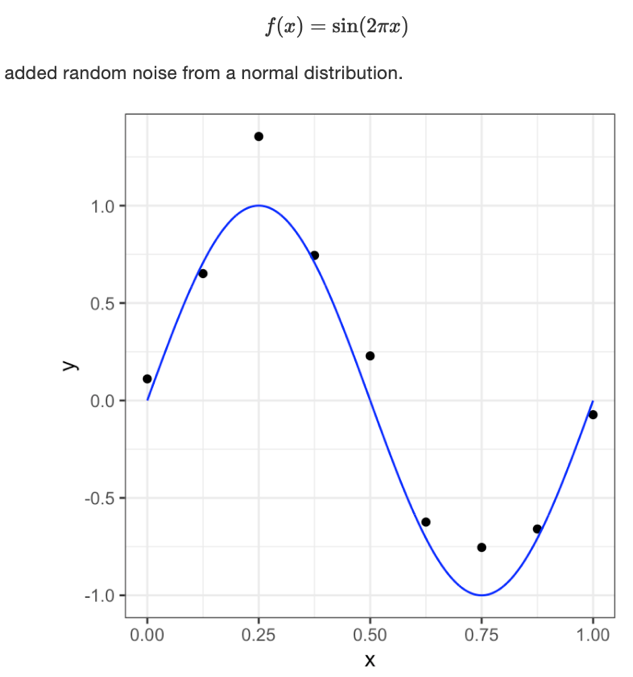
x = features we use to predict the outcome

🡪 To build a model that provides a prediction for any set of observed values x1,x2,...x5, we **collect data for which we know the outcome**

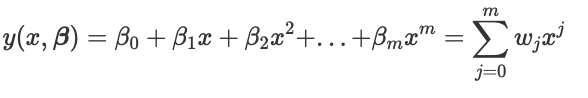
🡪 Use this data to train the model, use the trained model to apply to our new data for which we do not know the outcome

Avoid:

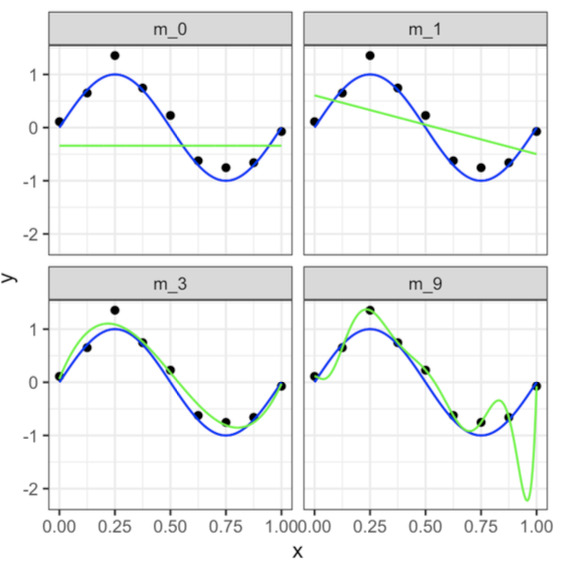
1. **Under-fitting**: the model does not capture the trends of the data resulting in a high measured error between actual and predicted outcome
2. **Over-fitting**: the model does not generalize well but fits the data used for training the model too well 🡪 becomes less severe as size of data set increases

Polynomial Curve fitting Example:

* Goal: predict y for some value of x
* Exploit dataset which involves discovering the underlying function 🡪 have to **generalize** **from a finite dataset of only n points**
* Polynomial function:



* Polynomial coefficients = b
* Polynomial function y(x,b) is a **nonlinear function of x** but it is a **linear function of the coefficents b 🡪 Linear regression model 🡪 Minimizing sum of squared errors (LSE)**

****

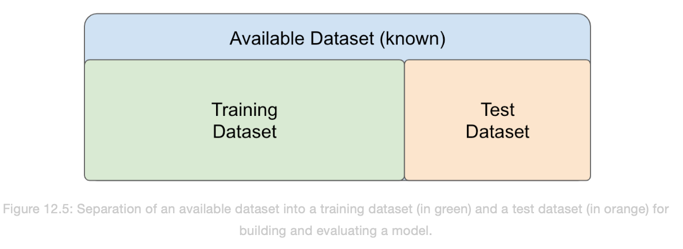
* Polynomial order = m

m = 0 & 1: under-fitting: fail to capture trend, high error between predicted & actual data

m = 3 best fit

m = 9 considers each deviation in the data points (incl. noise); too sensitive; poor generalization to other datasets is expected: over-fitting

**Generalization error = Expected error on unseen data**

* key issue: easy with flexible mathematical functions to fit extremely well to a dataset
* challenge: **reducing the error on unseen data**
* **How** to minimize error on data we have never seen?:
  + Assumption: **observations** x of our dataset and of **unseen data** are **i.i.d**, they are independent observations of the same population
  + 🡪 select subset of our dataset and pretend it is an independent dataset
  + Ein Bild, das Text enthält.

    Automatisch generierte Beschreibungdivide the available dataset into a training set and a test set.
  + **train** our algorithm exclusively on the **training set**
  + **test set** only for **evaluation** purposes **(10-30% of data)**

**Overfitting of the training set** = measured **error** computed from a defined error function is notably **larger** **for the test dataset** **than for the training dataset**.

## **Ein Bild, das Tisch enthält. Automatisch generierte BeschreibungCross-validation**

= strategy to assess the performance of a machine learning model help to **prevent over-fitting**

* data is randomly split into k folds: k=3:
* then model trained on k-1 folds and evaluated on fold not used for training
* high k: data training better imitates original data set, but slower computation time; k = 5 & k = 10 usually in practice
* i**.i.d assumption**: training & test samples have to be i.i.d for cross-validation to work
  + non-iid arises when data comes from clusters we don’t know

**Cross-validation in R:**

* library(**caret**)
* **k**<- 5 # number of folds
* **1**: Convert **Outcome** into **string: 1 = Yes, 0 = No**
* **2:** Validation specification:
  + - fitControl <- **trainControl**(method = "cv", *# cv for cross-validation*

number = k,

classProbs=TRUE, *# compute class probabilities*

summaryFunction = twoClassSummary)

* **3:** Train the model:
  + lr\_fit <- **train**(*## formula and dataset definition* 
    - * sex~.,  
        data = heights\_dt,  
        *## model specification*method = "glm",  
        family = "binomial",  
        *## validation specification* trControl = fitControl,  
        *## Specify which metric to optimize* metric = "ROC")
* get performance measurements: lr\_fit$resample
* get final model: lr\_fit$finalModel
* Fold with highest AUC:
  + metrics\_dt <- as.data.table(lr\_fit$resample)
  + metrics\_dt[order(-ROC)]

**Random Forests as alternative models**

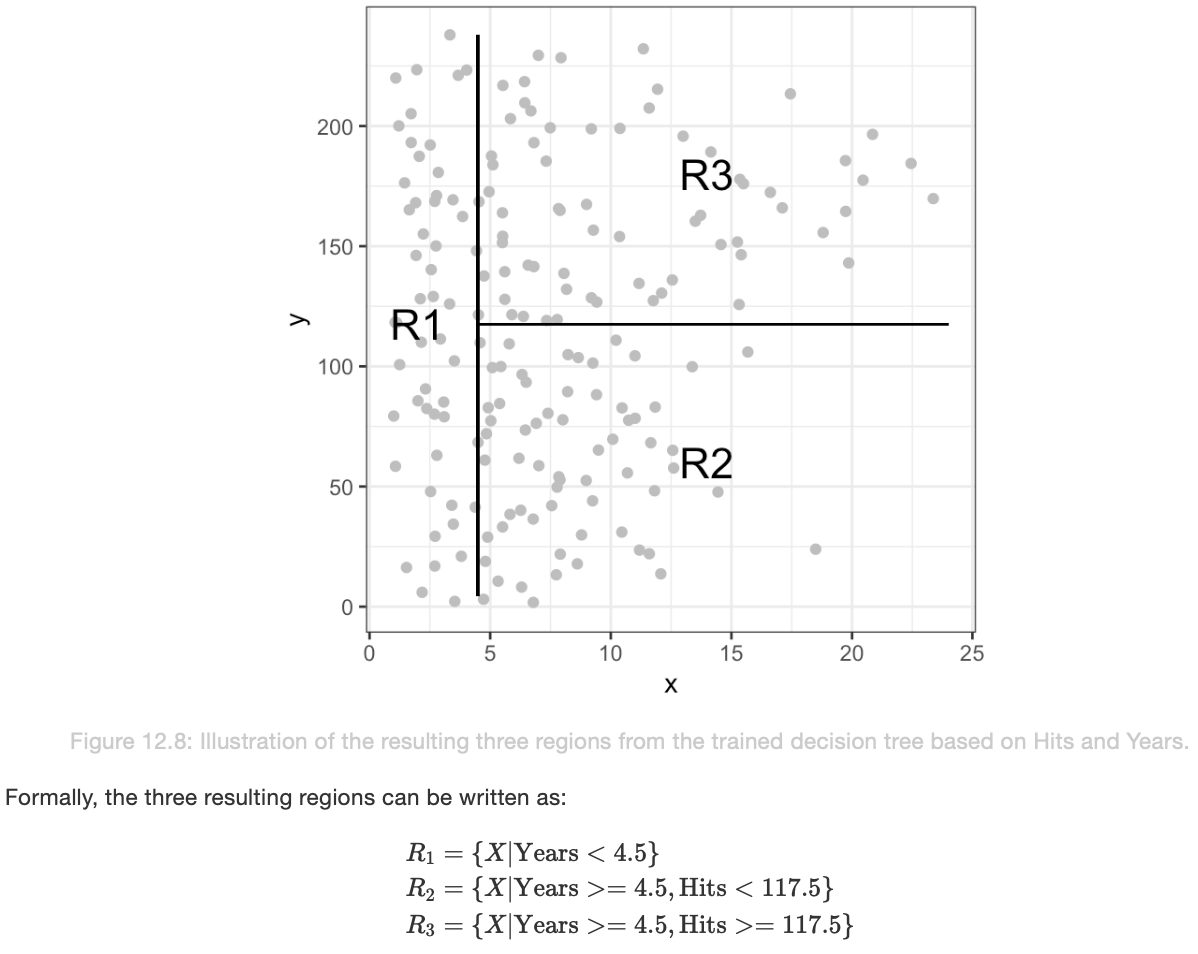
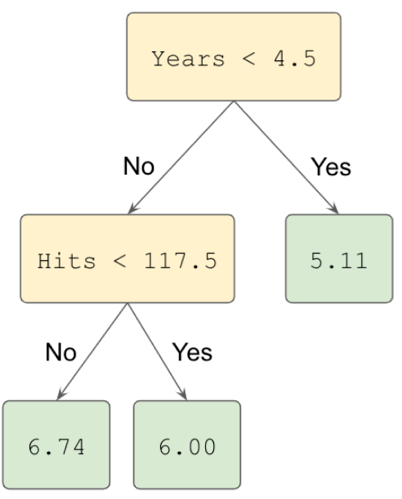
* non-interpretable supervised learning model
* applied to both regression and classification tasks
* based on an instance of tree-based ensemble learning 🡪 robust to over-fitting and allows fitting flexible functions

## Decision tree:

partitioning/ segmenting the training data set into a number of simple regions

**Decision tree for regression tasks:**

* Example: predicting log salary of players based on features years and hits
* the predicted outcome for an observation is given by the mean response of the training observations that belong to the same leaf node
  + 🡪 3 regions for players



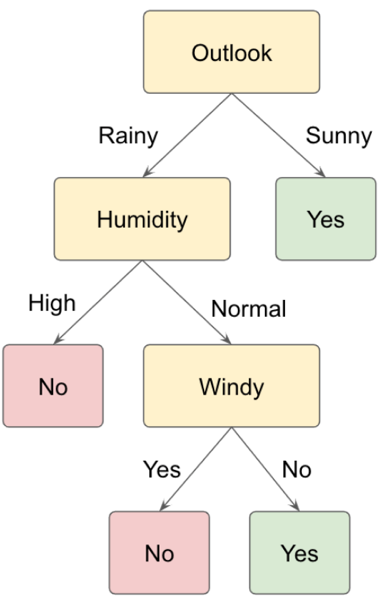
🡪 Years most important feature: Players with less experience earn lower salaries

🡪 If a player is more experienced, the number of hits that he made in the previous year is important for determining his salary

**How to find right threshold?**

* **Goal of decision tree** for regression tasks: optimal regions that **minimized residual sum of squares (RSS)**
* Top-down approach with recursive binary splitting 🡪 best reduction in RSS
  + **Ein Bild, das Text, Uhr enthält.

    Automatisch generierte Beschreibung**Minimization of:
* Stopping criteria:
  + Set a minimum number of samples in a region or maximum number of resulting regions

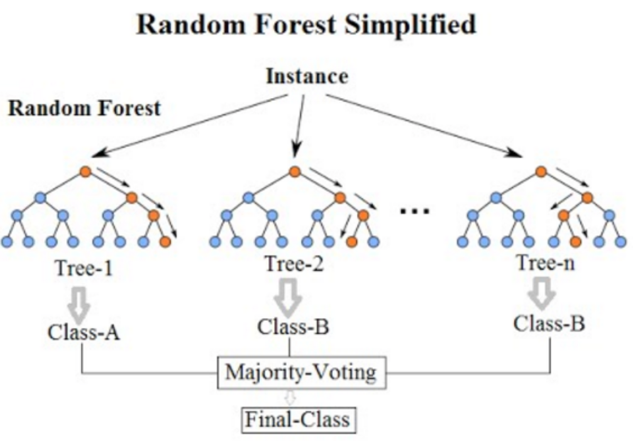
**Decision tree for classification tasks:**

* predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs (<-> regression task)
* **Cross-entropy** used for tree-splitting (<-> regression task: RSS)
* 3 features here
* Can be both numerical and categorical features

**Decision Trees in R:**

* Build decision tree (classifier):
  + dt\_classifier <- **rpart**(y ~x + x…, data =diabetes\_dt, diabetes\_dt , control = **rpart.control**(minsplit = 3, cp = 0.001))
* diabetes\_dt[, **deci\_tree\_pred** := predict(dt\_classifier, **type** = "**prob")[,2]]**
* For Training and Testing:
  + smp\_size <- floor(**0.70** \* nrow(data\_dt)) 🡪 70% Training Set
  + set.seed(13)
  + train\_ind <- sample(seq\_len(nrow(data\_dt)), size = smp\_size)
  + data\_dt[train\_ind, dataset:="train"] 🡪 label train & test data set
  + data\_dt[-train\_ind, dataset:="test"]
  + train on training dataset:
    - dt\_classifier <- rpart(y ~x+x…, data =data\_dt[train\_ind], control = rpart.control(minsplit = 3, cp = 0))
  + get predictions for both train and test set:
    - data\_dt[, preds\_dt := predict(dt\_classifier, type="prob", newdata=data\_dt)[,2]]
  + Ready for ROC PLOT!

## **Random Forests for classification and regression tasks**



constructing a random forest:

1. Step: aggregation (=bagging): generating many predictors, each using regression or classification trees.

- ensure that individual trees are not the same: bootstrap to induce randomness.

🡪 **bootstrap** makes the individual trees randomly different,

🡪 combination of trees is the forest

Random forest for **classification** tasks: the predicted output is the majority vote of the contained decision trees

Random forest for **regression** tasks: the predicted value is the mean of the predictions from each decision tree in the random forest

*Specific Steps:*

1. Build *B* decision trees using the training set. Fitted models as *T*1, *T*2, . . . , *TB*.
2. For every observation in the test set, form a prediction *y*^using tree *Tj*
3. continuous outcomes, form a final prediction with the average *y*^ = *1/B (Sum y^bis B)*
   1. categorical outcomes, predict *y*^ with majority vote (most frequent class among *y*^1, . . . , *y*^*B*.

*Inducing randomness:*

* + Create a **bootstrap** **training** set by sampling N observations from the training set with replacement 🡪 each tree is trained on a different dataset.
  + Randomly select a subset of the features to be included in the building of each tree so that not all features are considered for each tree. A different random subset is selected for each tree. Reduces correlation between trees in the forest and prevents over-fitting.

**Random Forests in R**

* library(randomForest)
* rf\_classifier <- **randomForest**(*## Define formula and data*

sex~.,

data=heights\_dt[train\_ind],

***## Hyper parameters*****ntree**=100, *# Define number of trees*

**nodesize** = 5, *# Minimum size of leaf nodes*

**maxnodes** = 30, *# Maximum number of leaf nodes*

**mtry = …,**

**sampsize = length(train\_ind)**

*## Output the feature importances*

**importance**=TRUE)

* Predict on all data:
  + data\_dt[, preds\_rf := predict(rf\_classifier, type="prob", newdata=data\_dt)[,2]]
* rf\_classifier$importance

*Hyper-parameters: to be tuned in each application*

* Ein Bild, das Tisch enthält.

  Automatisch generierte Beschreibung height of the person is the most important feature for predicting gender for both females and males
* the mother’s height to be more important for predicting the female gender than the father’s height
* the father’s height to be more determinant for predicting the gender male than the mother’s height
* Predicting the gender of an input sample using the trained model stored in variable rf\_classifier 🡪 predict()
  + heights\_dt[, sex\_predicted:**= predict**(rf\_classifier, heights\_dt[,-**c**("sex")])]

**# lapply returns list**

**# sapply tries to concatenate everything (vector, matrix)**