# D1 Modul - Handling and Visualising Data





## Databases

Problem with flat-files:

1. Structural relationships are missing
2. The files are redundant, the same information re-appears in a file or throughout a set of files

Solution: data model and data base (DB)

DB management system (DBMS)

## Principles of Relational Databases (RDBs)

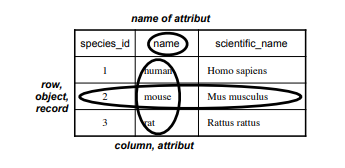
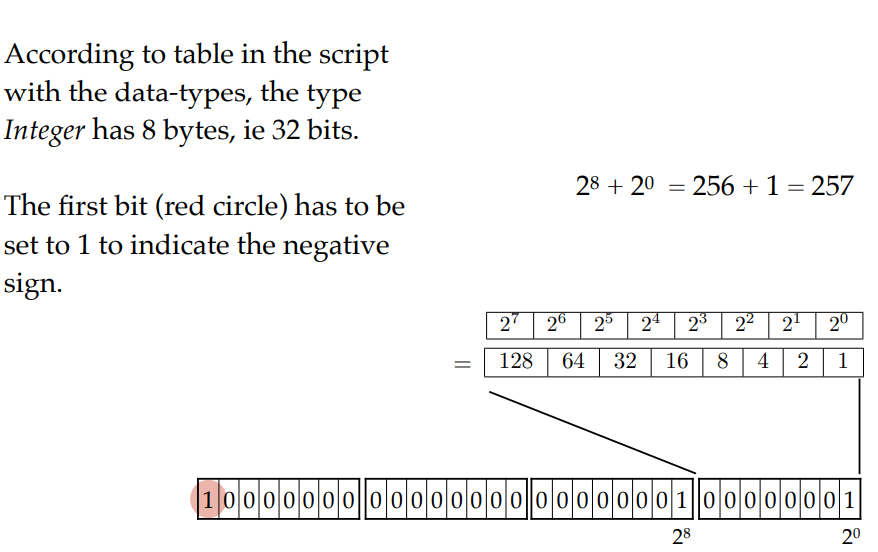


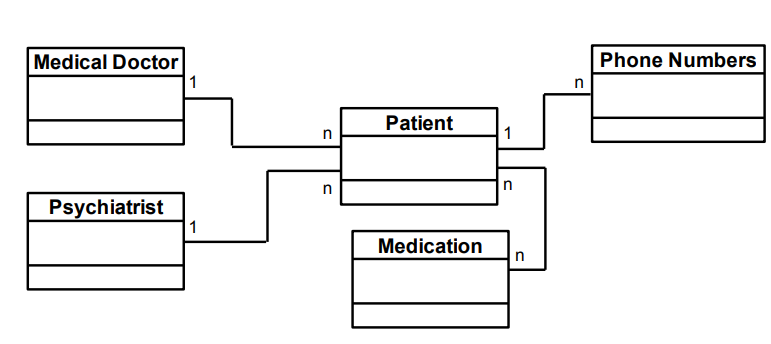
Figure 1: Tables are organized in columns and rows. The columns represent different attributes that you want to store. You can have, for example, a table Species with columns representing the common name, scientific name, and some id

Data types:

Exercise 2.1.1 How would the number -257 be represented internally in the computer as data type Integer.

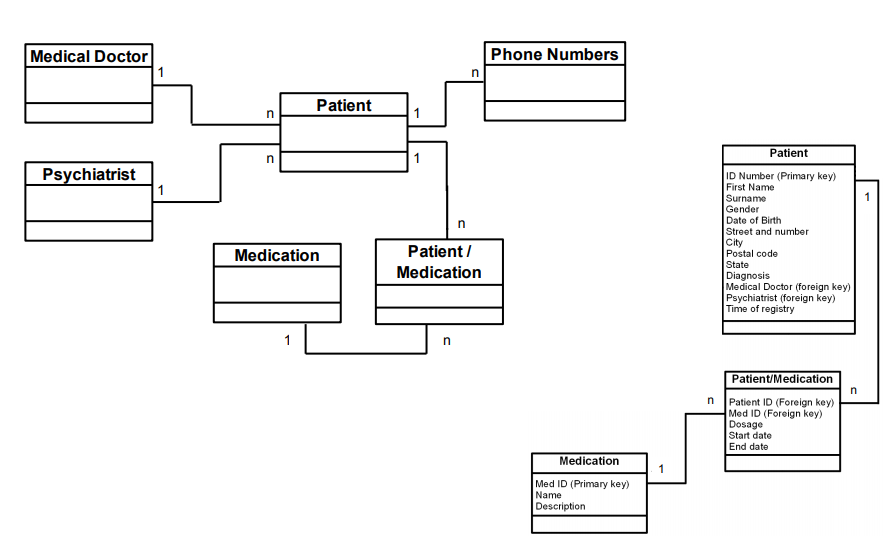


Exercise 2.1.2 Consider the following part of a mental health clinic DB.

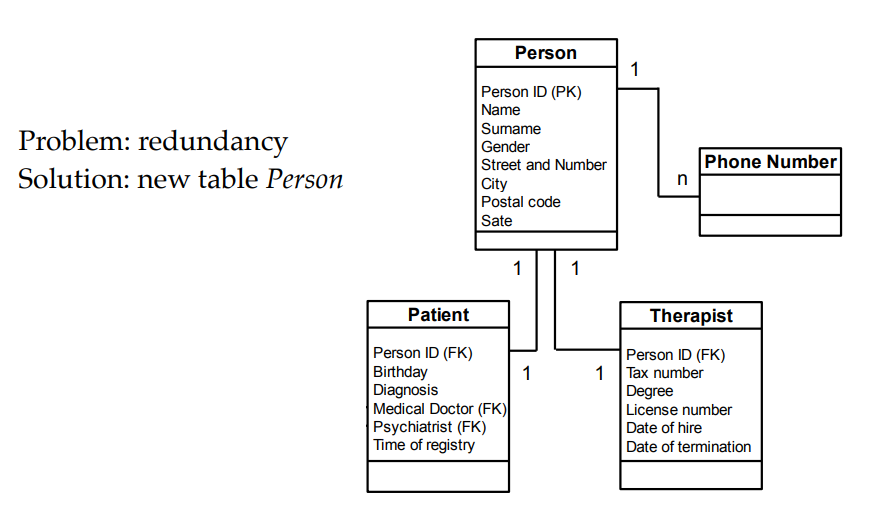


Exercise 2.1.3 Here is a detailed view on the tables patient and medication.

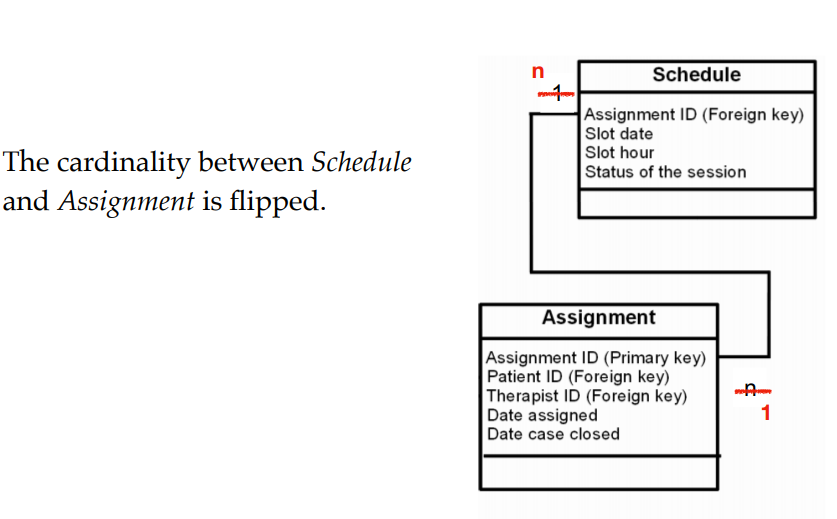
The relation between the two is many to many. As mentioned in the section about cardinalities, tables that are related by many to many can not be connected directly. An intersection table is needed to connect them. Add an intersection table between patient and medication, name it Patient/Medication connect it to patient and medication using keys and add the attributes Dosage, Start date and End date.



Exercise 2.1.4 Now you can look at Figure 2.1. Study the entire design and discuss it with your peer(s). Would there be a problem with the design, if a therapist can also be a patient?



Exercise 2.1.5 There is one mistake ”hidden” in Figure 2.1 concerning cardinalities. Can you find it?



## 2.1.2 Querying RDBs

**Apache OpenOffice** is an open-source office suite, i.e. a collection of bundled software, including a word processor, a spreadsheet, a presentation program and more. It also contains Base, a relational database management system (analogous to Microsoft Access).

**Task 1:**

Load the database toyGeneDB.odb in OpenOffice. Double clicking on the file will not work on some systems; instead, it has to be opened from within OpenOffice

Hint: 1. Start OpenOffice; 2. Click on Database; 3. Check the radiobutton Open an existing database file; 4. Click Open and navigate to the location of toyGeneDB.odb on your computer.

c. What is their cardinality?

d. How are the tables related? Hint: In the menu-bar, click on Tools and then Relationships.



In the following tasks you will create queries using the Design View. The solutions to all the queries can be found in the database toyGeneDB withQueries.odb. Task 2 a. Query1: Select the column name from the table protein (hereafter, referred to as protein.name).

Task 2

Query 1

Select the column "name" from the table "protein" (hereafter, referred to as "protein.name")

SELECT "protein"."name" FROM "protein"

Query 2

Select the columns name and sequence from the table protein, sort the result in descending order of the names (use "Sort" in the design view)

SELECT "name" FROM "protein" ORDER BY "name" ASC

Query 3

In the species table, show the name and scientific\_name of the rat

Hint: use "Criterion" in the design view

SELECT "name", "scientific\_name" FROM "species" WHERE "name" = 'rat'

Query 4

Do the same as in Query 3, but only display the scientific\_name

SELECT "scientific\_name" FROM "species" WHERE "name" = 'rat'

Query 5

In the species table, show the name and scientific\_name of rat and human

Hint: use "Criterion" and "Or" in the design view

SELECT "name", "scientific\_name" FROM "species" WHERE ( "name" = 'rat' OR "name" = 'human' )

Query 6

Select the protein names and sequences for proteins that contain the subsequence SHS.

Hint: write the following in the appropriate "Criterion"-field: LIKE 'SHS\*'

SELECT "name", "sequence" FROM "protein" WHERE "sequence" LIKE '%SHS%'

Query 7

Select the protein names and sequences for proteins that contain the subsequence "SHS" or the subsequence "GY"

SELECT "name", "sequence" FROM "protein" WHERE ( "sequence" LIKE '%SHS%' OR "sequence" LIKE '%GY%' )

Query 8

Show all the human proteins

Hint: In the design view form, you need on column for the species table (to insert "human" in "Criterion"), and one column for the protein table (to select the protein name)

SELECT "species"."name", "protein"."name" FROM "protein", "species" WHERE "protein"."species\_id" = "species"."ID" AND "species"."name" = 'human'

Query 9

Count the number of human proteins

Hint: 1. Build on Query 8; 2. make sure to disable visibility of the species name; 3. Use the function "Count"

SELECT COUNT( "protein"."name" ) FROM "protein", "species" WHERE "protein"."species\_id" = "species"."ID" AND "species"."name" = 'human'

Task 3

SQL-Query 1

Modify Query 1 to show species.scientific\_name

Solution: SELECT "scientific\_name" FROM "species"

SQL-Query 2

Modify Query 2 to sort in descending order (substituting DESC by ASC)

Solution: SELECT "name", "sequence" FROM "protein" ORDER BY "name" ASC

SQL-Query 3

Modify Query 3 to show the scientific name (only) of human

Solution: SELECT "scientific\_name" FROM "species" WHERE "name" = 'human'

SQL-Query 4

Modify Query 5 to show the sequence of "GTM1\_HUMAN" and "GTM2\_HUMAN"

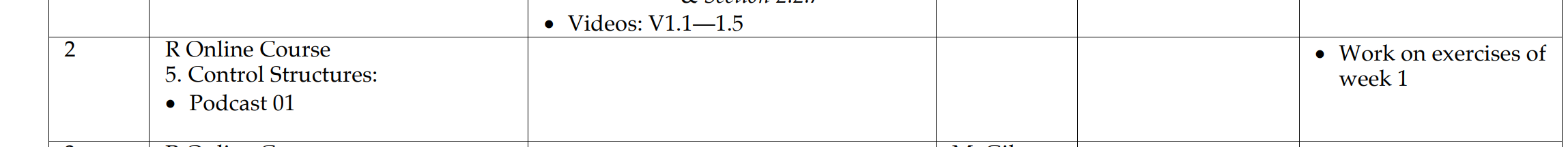
Solution: SELECT "sequence" FROM "protein" WHERE ( "name" = 'GTM1\_HUMAN' OR "name" = 'GTM2\_HUMAN' )

SQL-Query 5

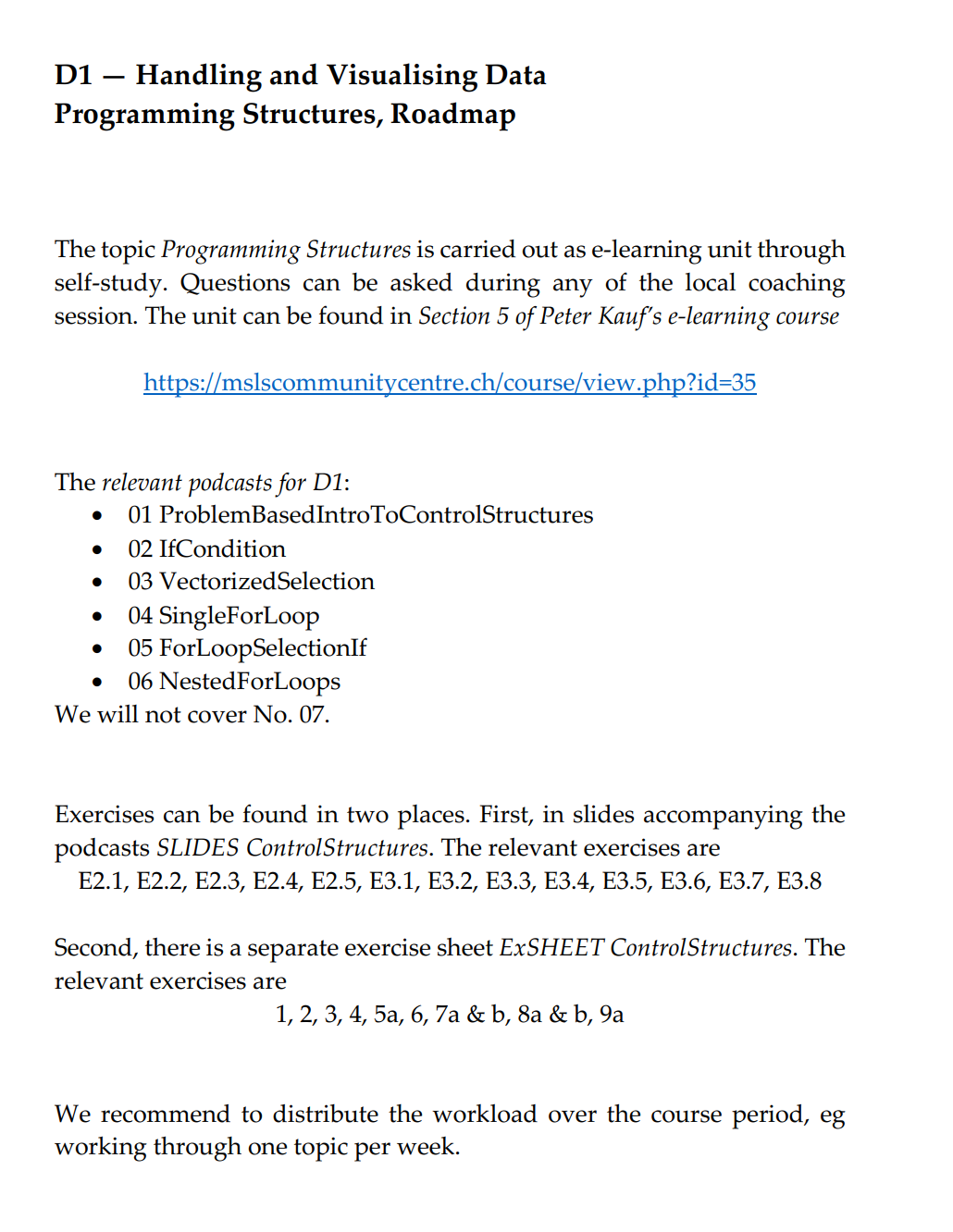
Modify Query 6 to show results of proteins that contain the subsequence KM

Solution: SELECT "name", "sequence" FROM "protein" WHERE "sequence" LIKE '%KM%'

## Week: 22. 09.20: Online Lecture, starts on 12:15 HH:MM



<https://mslscommunitycentre.ch/course/view.php?id=35>

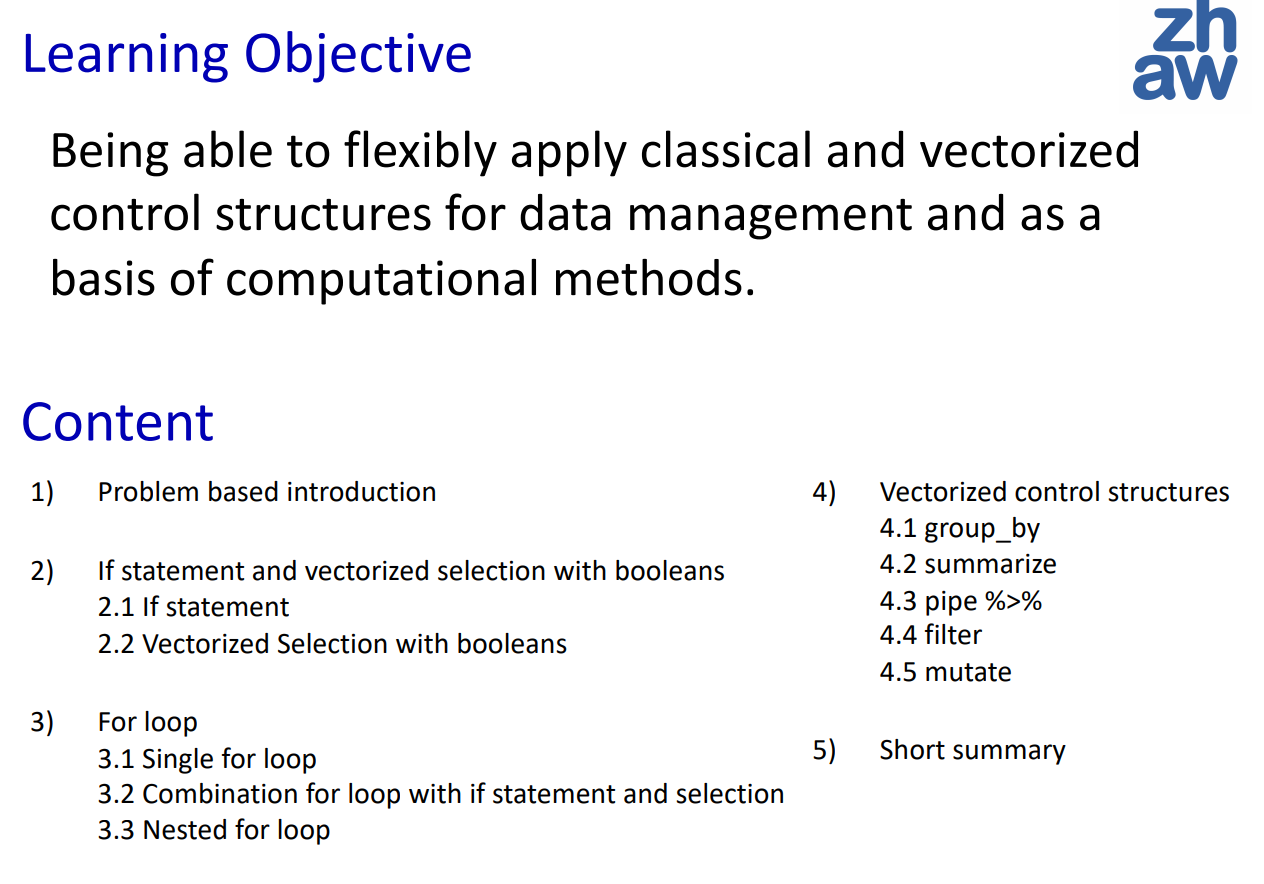


### [Section 5: Control Structures](https://mslscommunitycentre.ch/course/view.php?id=35#section-6)

* This section contains learning materials for control structures. It is organized as follows:
  + SLIDES: here you find slides with all the discussed topics and implementation exercices. The R-Code for the slides can be found in "RCode\_Slides\_ControlStructures.R".
  + Podcasts 01 - 07 explaining the topics, based on the SLIDES. Please note that Podcast07 is too large for moodle and needs to be downloaded through a link. The external repository may not be accessible from nets with too heavy security restrictions (in this case, download at home).  
    Please note: the podcasts are not "perfect" concerning resolution or sound quality. Their purpose is to give yet another dimension of explanation of the topics in this section.
  + [Data](https://mslscommunitycentre.ch/mod/resource/view.php?id=4173) as used in the SLIDES can be found in the zip folder "[Data](https://mslscommunitycentre.ch/mod/resource/view.php?id=4173)". The R-Scripts are set such that [data](https://mslscommunitycentre.ch/mod/resource/view.php?id=4173) is found if stored in a folder "[Data](https://mslscommunitycentre.ch/mod/resource/view.php?id=4173)" within the same folder as you store the R-Scripts.
  + ExSheet\_ControlStructures contains further exercices (based on the same [Data](https://mslscommunitycentre.ch/mod/resource/view.php?id=4173)). Solutions can be found in "[solutionsExerciceSheet](https://mslscommunitycentre.ch/mod/resource/view.php?id=4177" \o "solutionsExerciceSheet).R"

Warning: The content of this section may be a bit difficult. Please count for enough time to understand these important concepts. It is fully normal if you need more than one attempt at solving the exercices.

01 ProblemBasedIntroToControlStructures:





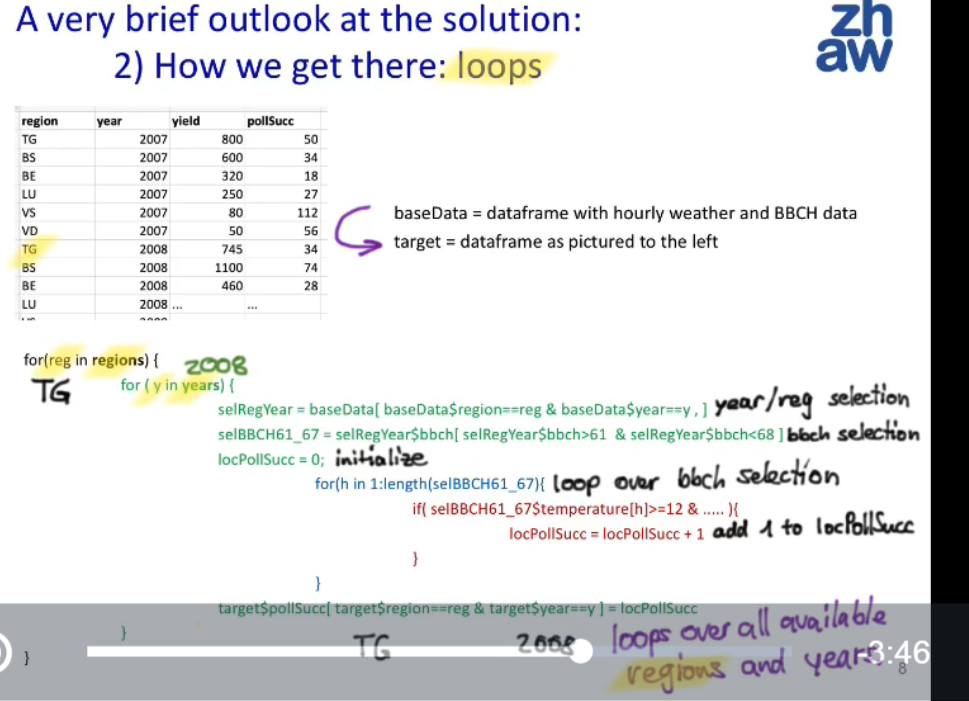
1. Problem based introduction to control structures:

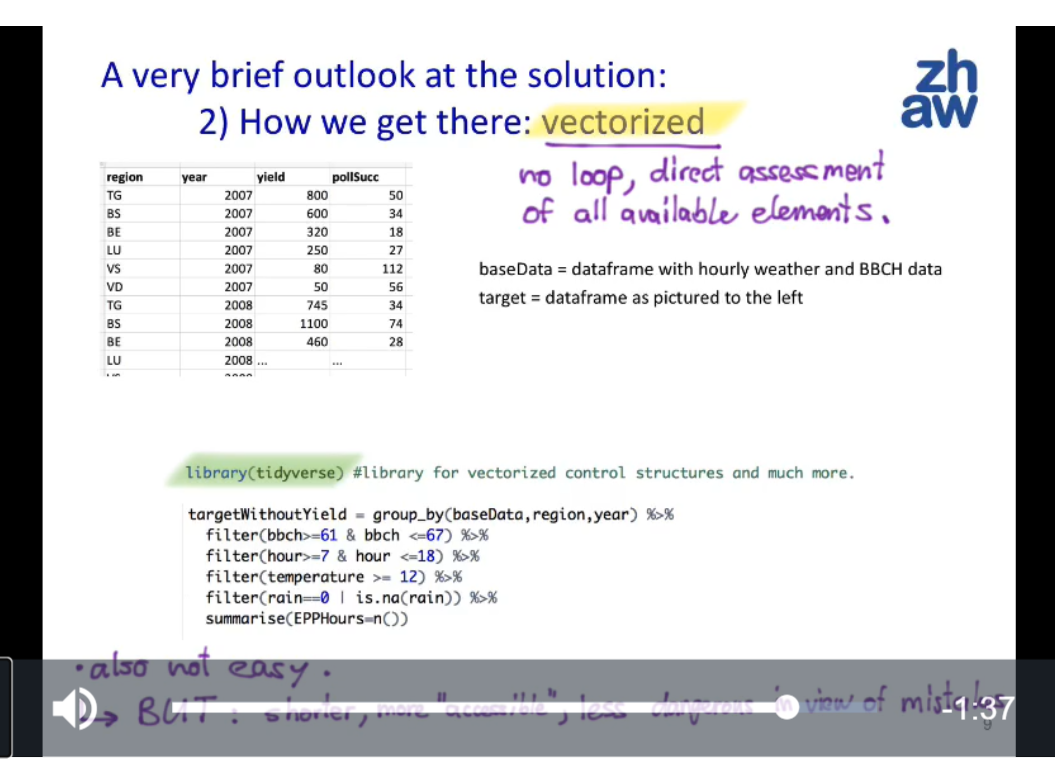


As we noted, algorithms require two important control structures: iteration and selection. The programmer can choose the statement that is most useful for the given circumstance.

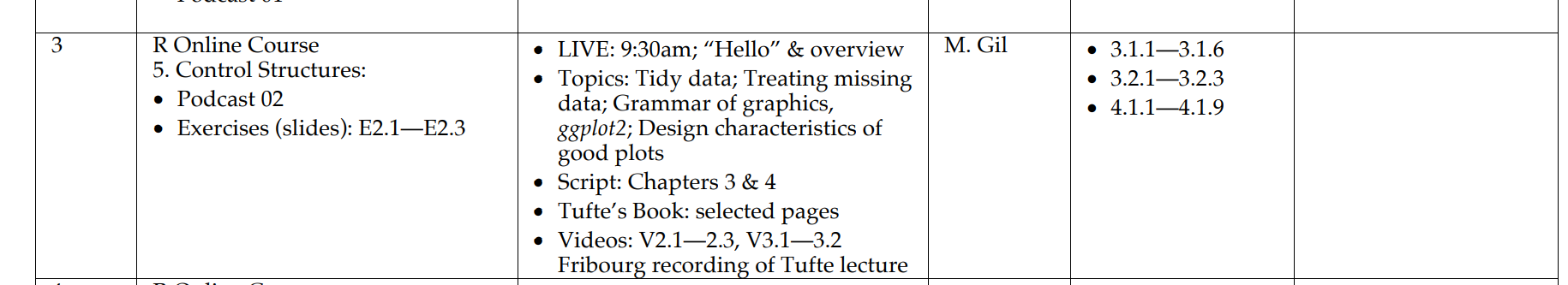
Iteration = loop and selection = if statement

Source: <https://runestone.academy/runestone/books/published/pythonds/Introduction/ControlStructures.html>

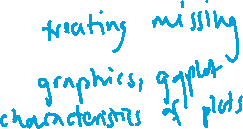




# Week

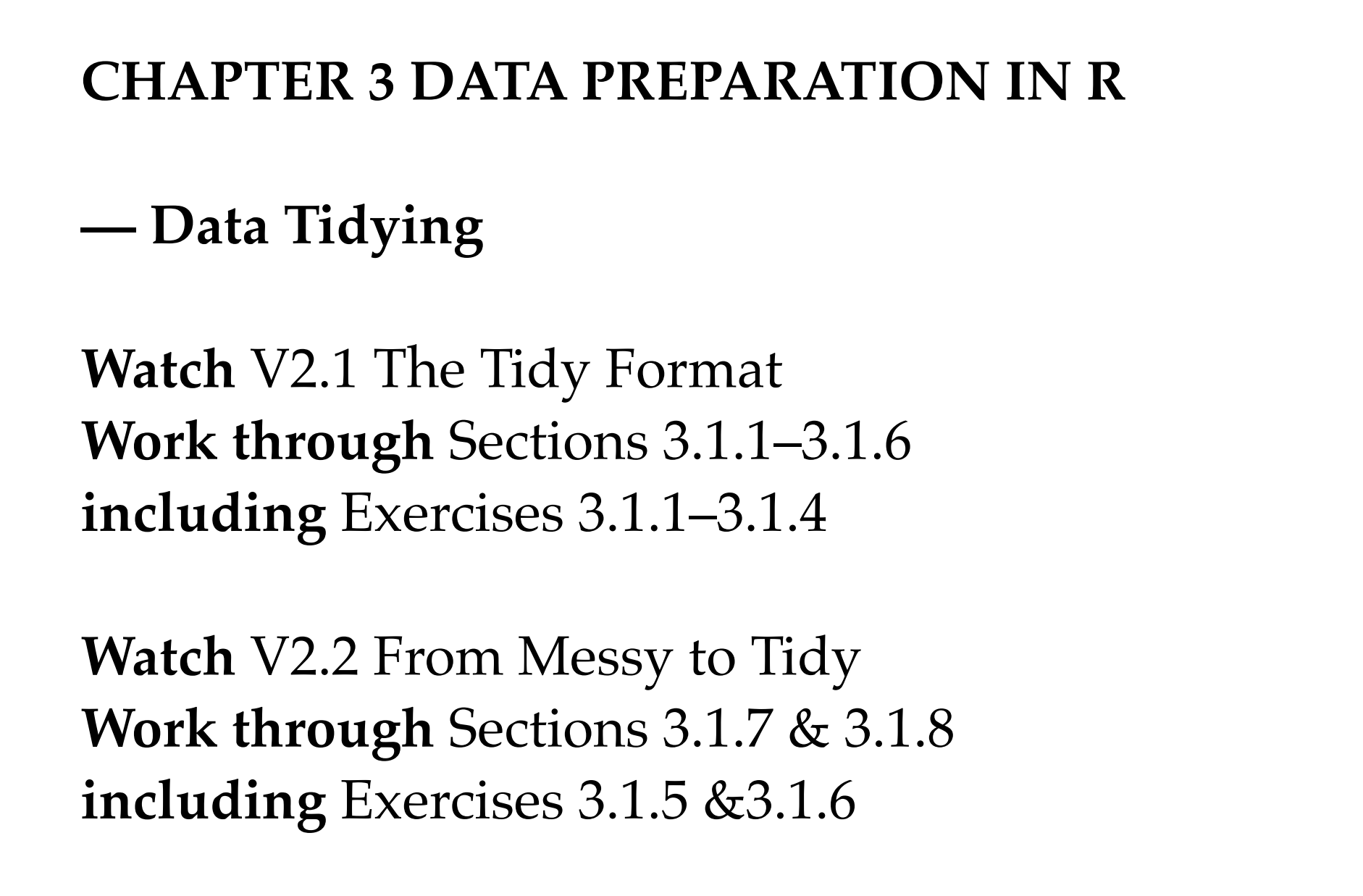


<

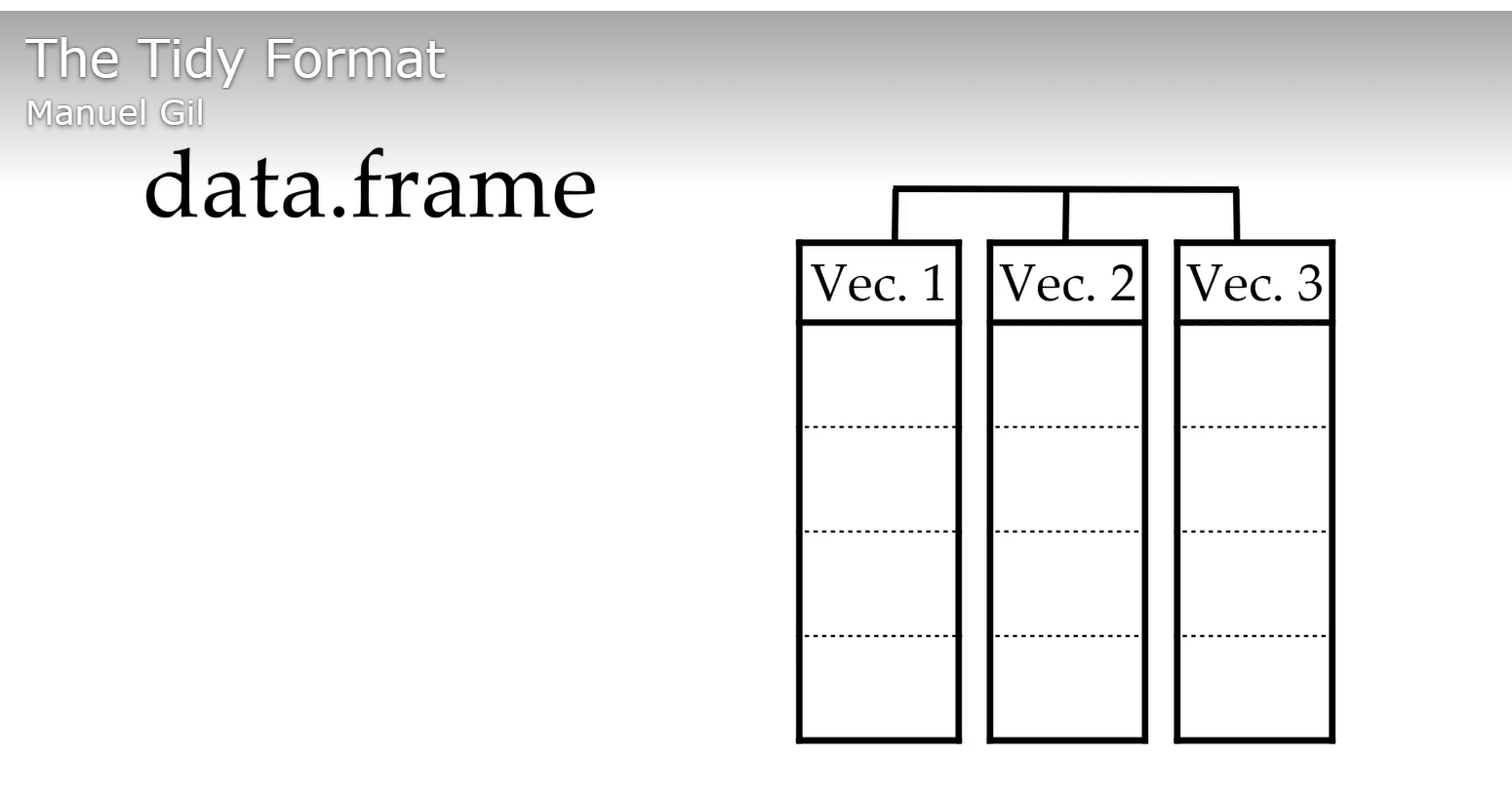


Data preparation and data visualisation

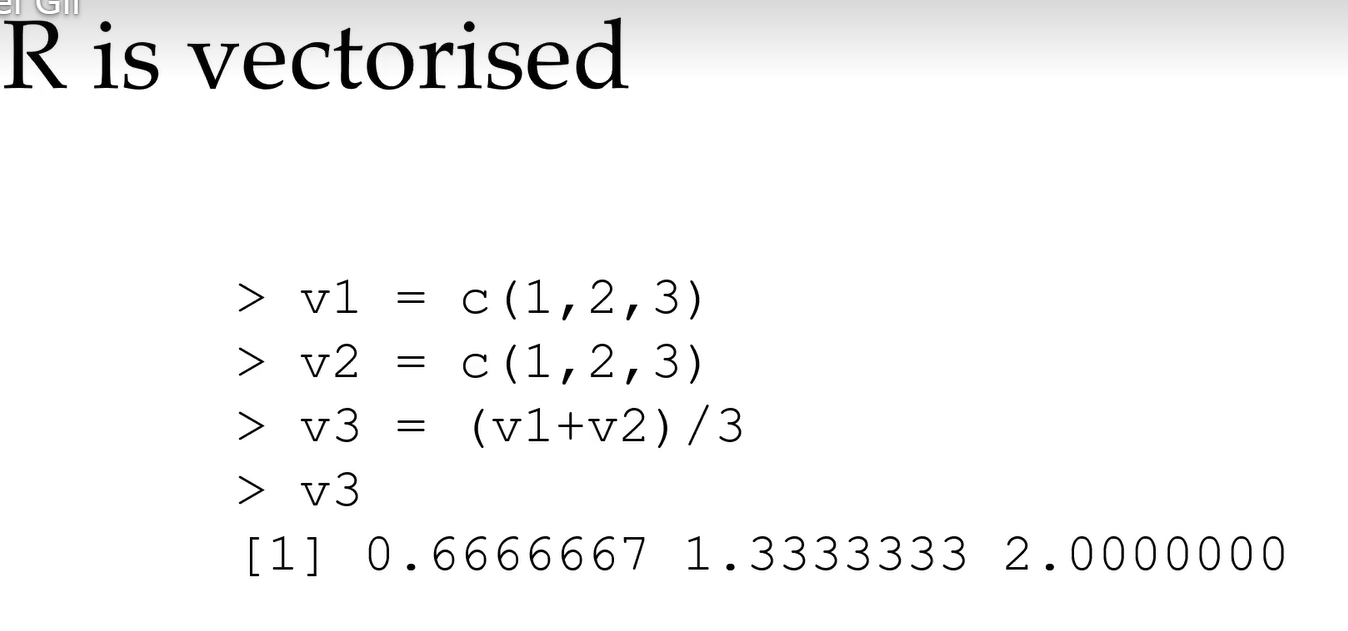
Data preparation: Detecting inconsistencies Handling missing data Outlier checking Proper parsing of formats, Data tidying Handling missing data

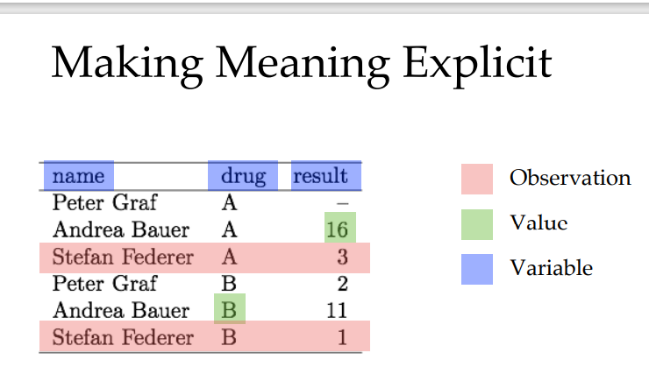






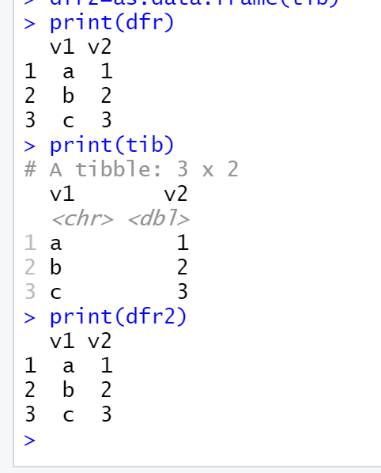




… a variable I.e. all values that measure the same attribute E.g. age, sex, eye colour, temperature, duration

… an observation I.e. all values measured on the same thing E.g a particular person, city, or experiment

## Week 3: Data Preparation – Exercises:

install . packages (" tidyverse ")

library ( tidyverse )

3.1.2 Data frames of type tibble



A tibble is a data frame. At any time, you can convert a tibble to R’s classic data frame and vice versa:

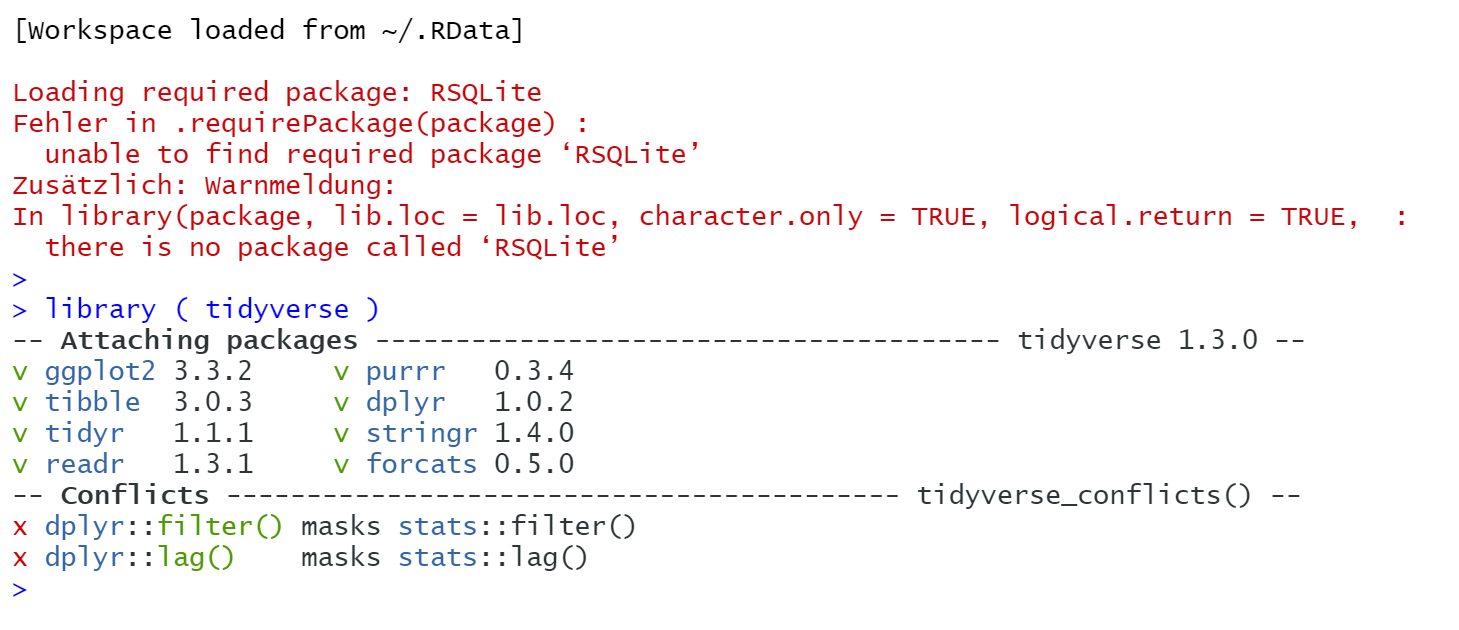


dfr = data . frame ( v1 =c("a", "b", "c") , v2 =c(1 , 2 , 3) )

tib = as\_ tibble ( dfr )

dfr2 = as. data . frame ( tib )

Note: In most cases tibbles behave like R’s classic data frames. One key difference between the two is printing. In contrast to classic data frames, tibbles report the type of each column: The refined print() method for tibbles shows only the first 10 rows, and all the columns that fit on screen. This makes it easier to work with large data. Furthermore, you can control the number of rows (n) and the width of the display. Note that **width=Inf** will display all columns.



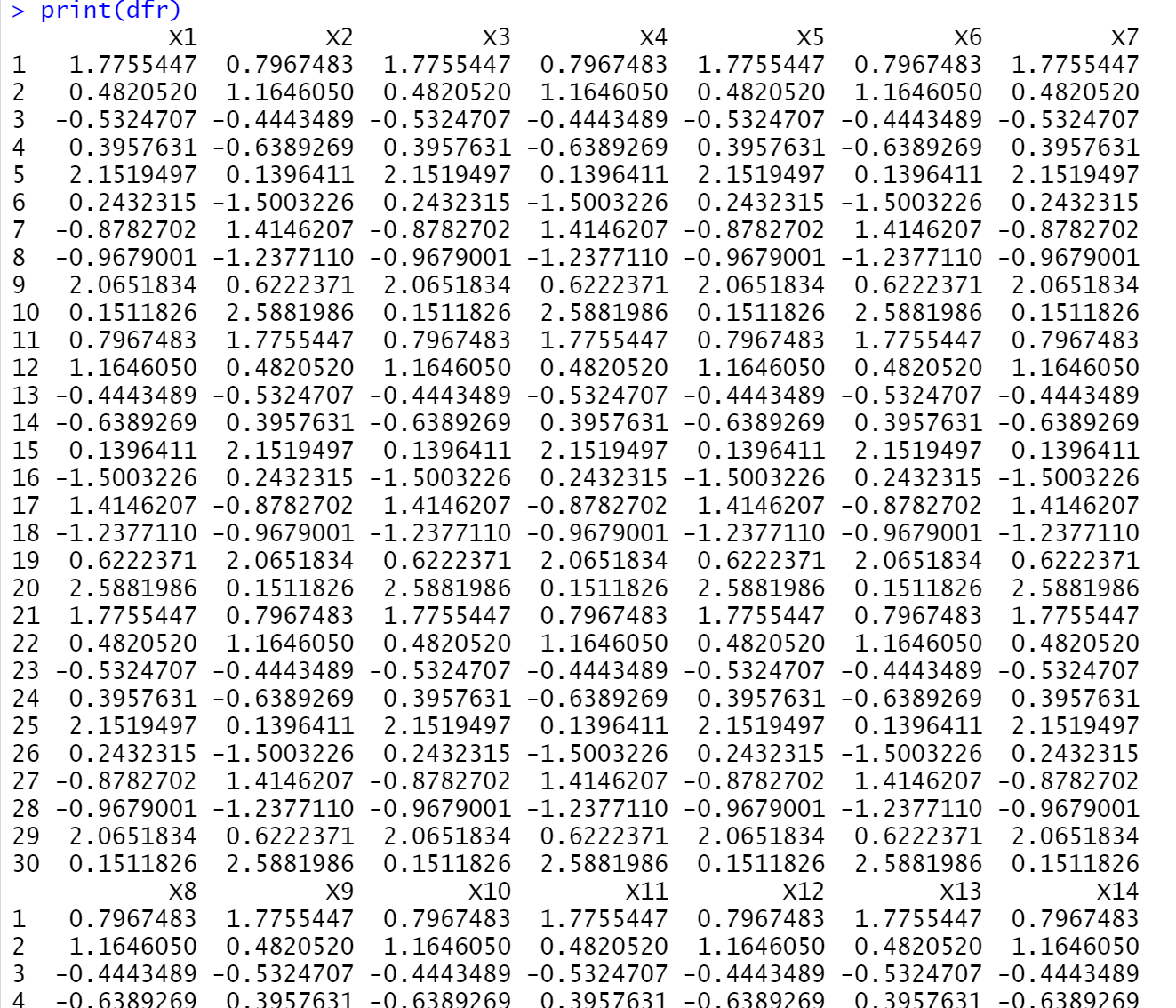
### Exercise 3.1.1 Experiment with the parameter n of the print() method.

### Exercise 3.1.2 Experiment with the parameter width.

# Data frame with normally distributed random sample :

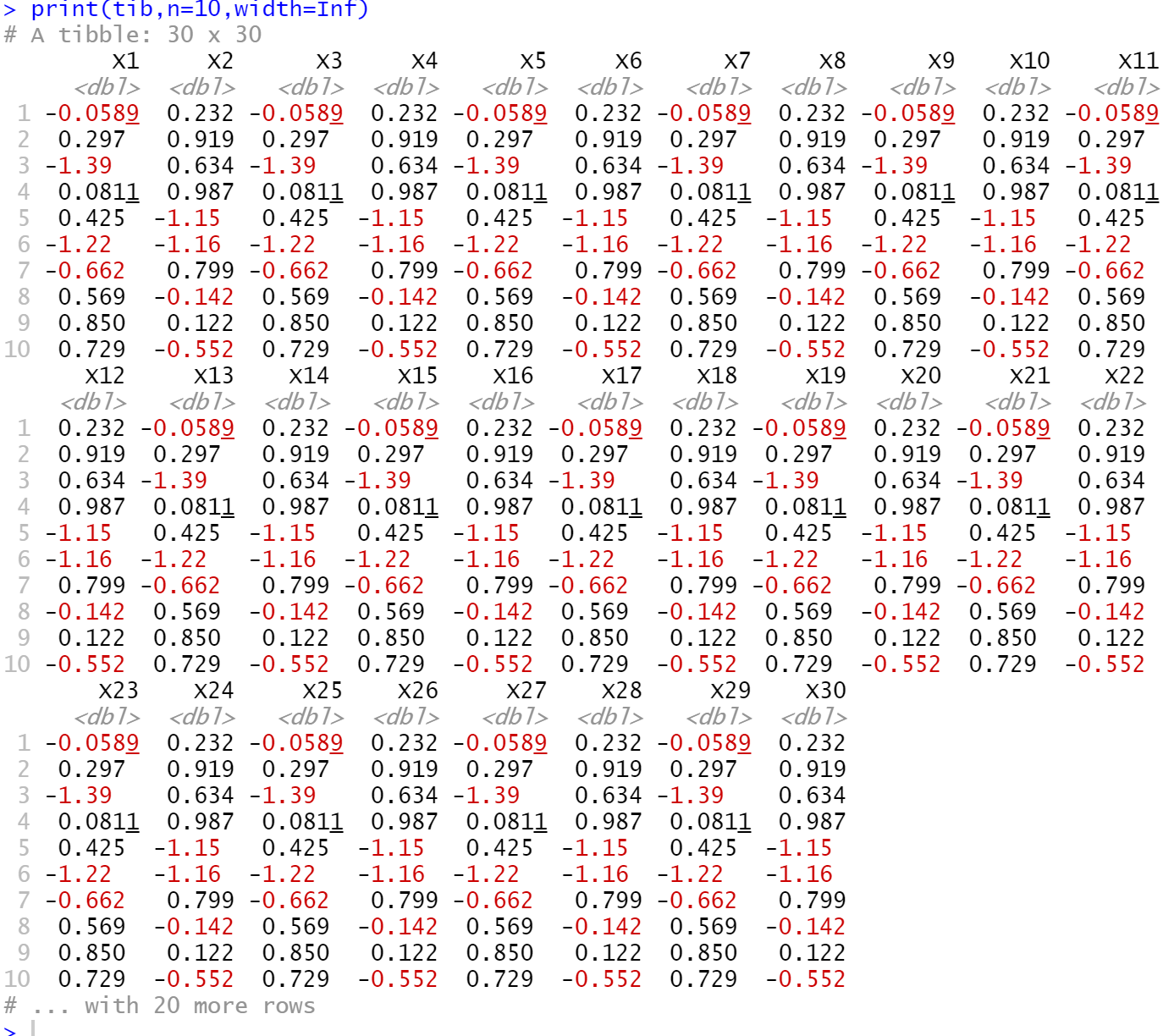
dfr = data.frame ( matrix(rnorm (20), ncol =30, nrow =30) )

print (dfr)



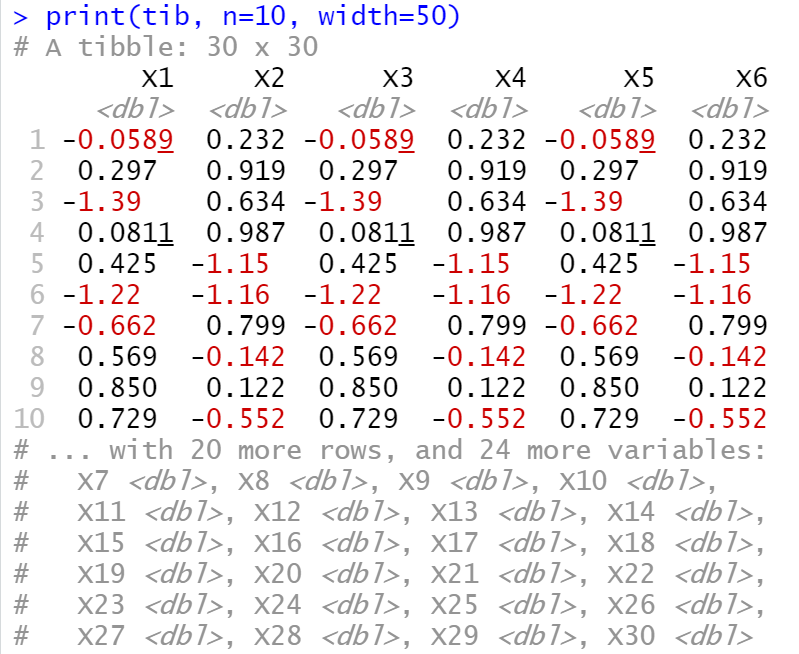
tib = as\_ tibble ( dfr )

print ( tib , n =10 , width = Inf )

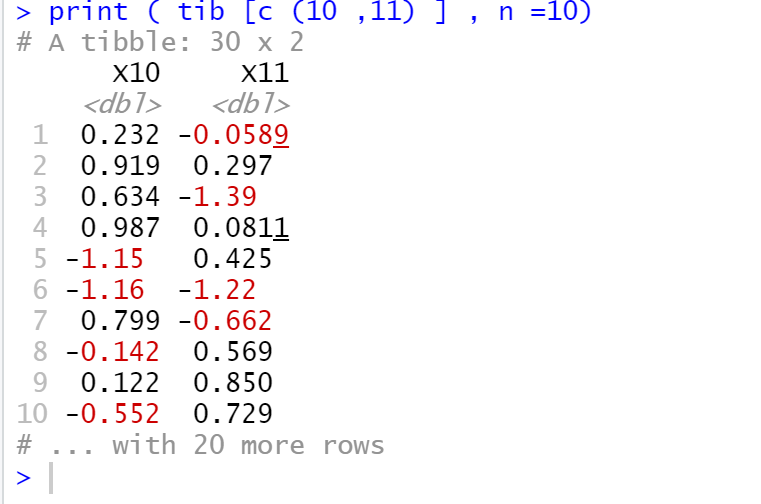




print ( tib , n =10 , width =50)



print ( tib [c (10 ,11) ] , n =10)

   
Alternatively, you can also work with the View() method:

View ( dfr )

View ( tib )

**Note** that is is possible for a data frame to have column names that are not valid R variable names, aka non-syntactic names. For example, they might not start with a letter, or they might contain unusual characters like a space. To refer to these variables, you need to surround them with backticks ‘:

dfr2 = data . frame ( ‘1 ‘=c("a", "b", "c") , ‘2 ‘=c(1 , 2 , 3) )

### Exercise 3.1.3 With the tidyverse package you have downloaded the following

### tables**: table1, table2, table3, table4a, and table4b.** They show the same dataset,

### but each table structures it differently. The data comes from the World Health

### Organisation, and records counts of confirmed tuberculosis cases. Inspect the

### tables and describe how the variables and observations are organised. Which

### one is the tidy table?

### Exercise 3.1.4 Using R’s vector arithmetic and the tidy table, compute for

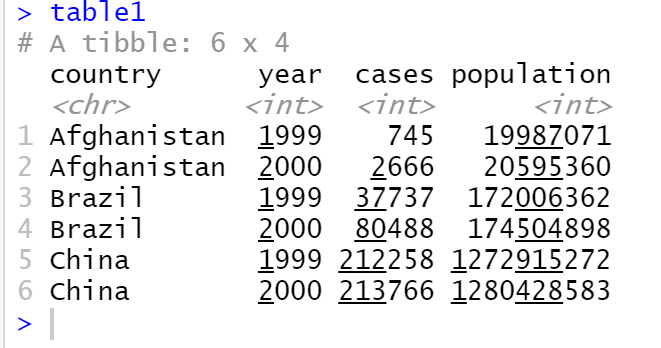
### each country the rate of tuberculosis cases (as cases per 100’000 inhabitants).

### Add the column to the tidy table and afterwards remove it again.

### **In the following, we will look at common problems with messy data.**

### # Solution Exercise 3.1.4

Table1



rate = table1$cases / table1$population \* 100000

rate

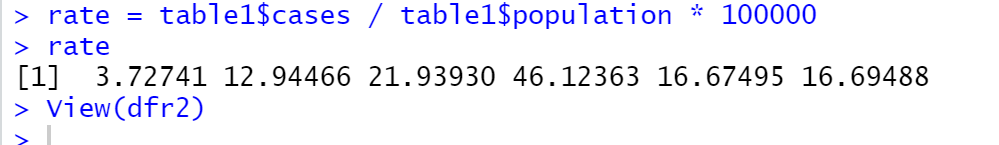


table1$rate = rate

table1

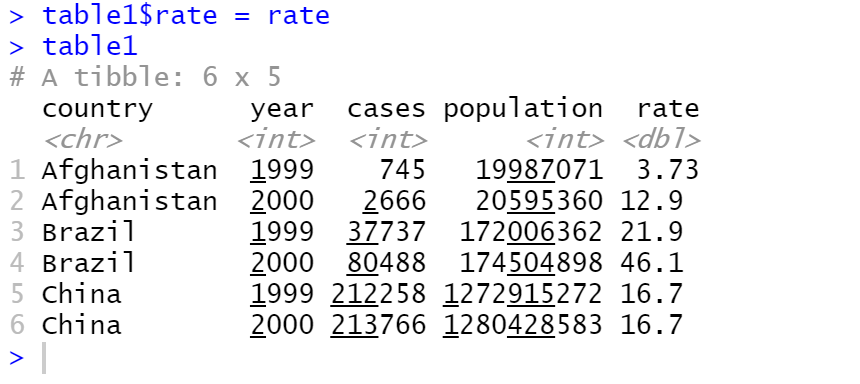
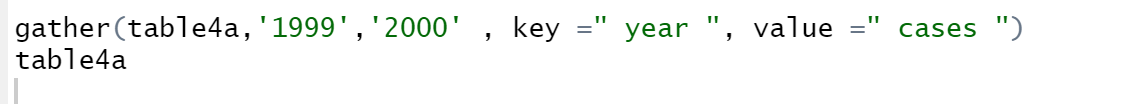


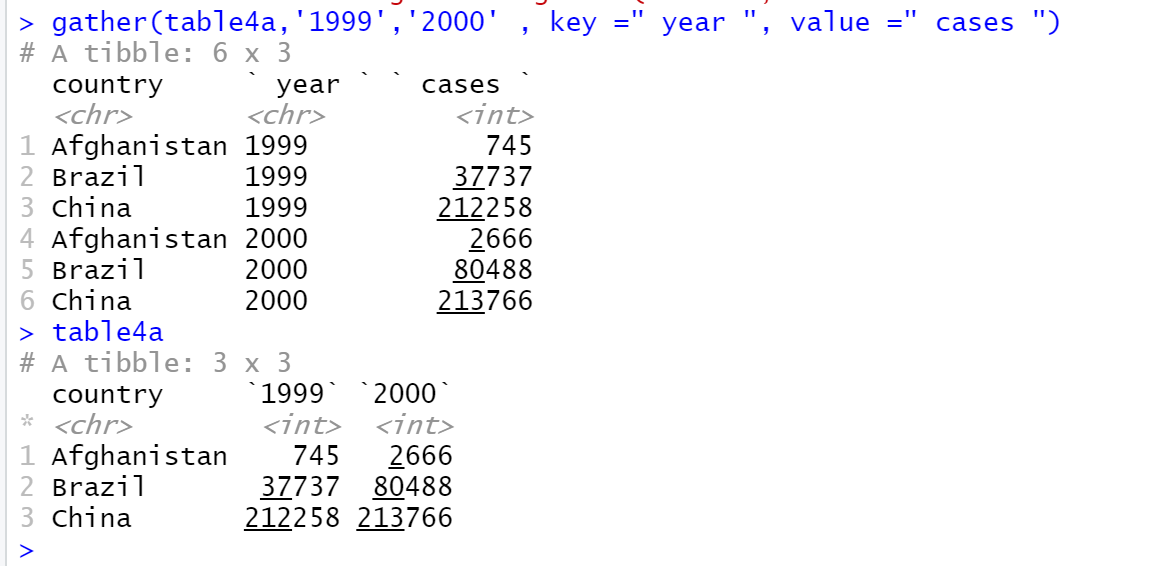
table1 = subset(table1, select=-rate)

table1

### Column headers are values, not variable names

In some datasets the column names are not names of variables, but values of a variable. This is the case for table4a. **The column names 1999 and 2000 represent values of the year variable, and each row represents two observations, not one.** To tidy a dataset like this, we need to convert those columns **into a new pair of variables**. This can be achieved with the function gather (), which needs the following information:

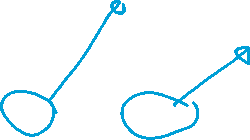


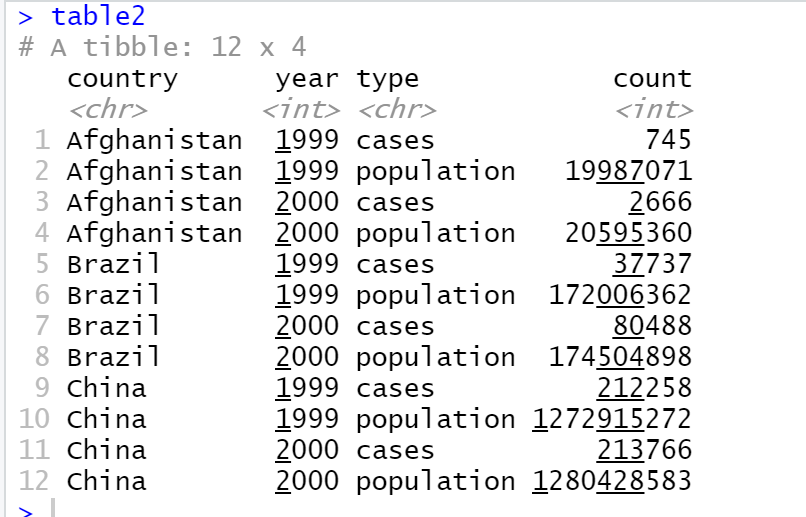


### Multiple variables are stored in one column

In this section we are concerned with **table2.** Note how the column count contains both the count of the tuberculosis incidents and the count of the population (i.e. population-size). The corresponding variable names are recorded in a column denoted with type. As a consequence, each observation is scattered across multiple rows. This can be fixed with the function spread(). It needs the following information:

* The column with the variable names (here, type). This column is passed by the parameter named key.
* The column that contains values forms multiple variables (here, count), passed by the parameter named value.





The call to spread() is:

spread(table2, key = type, value = count)

Exercise 3.1.5 Why does spreading this tibble fail? How could you add a new column to fix the problem?

people= tibble(

name=c("A", "A", "A", "B", "B"),

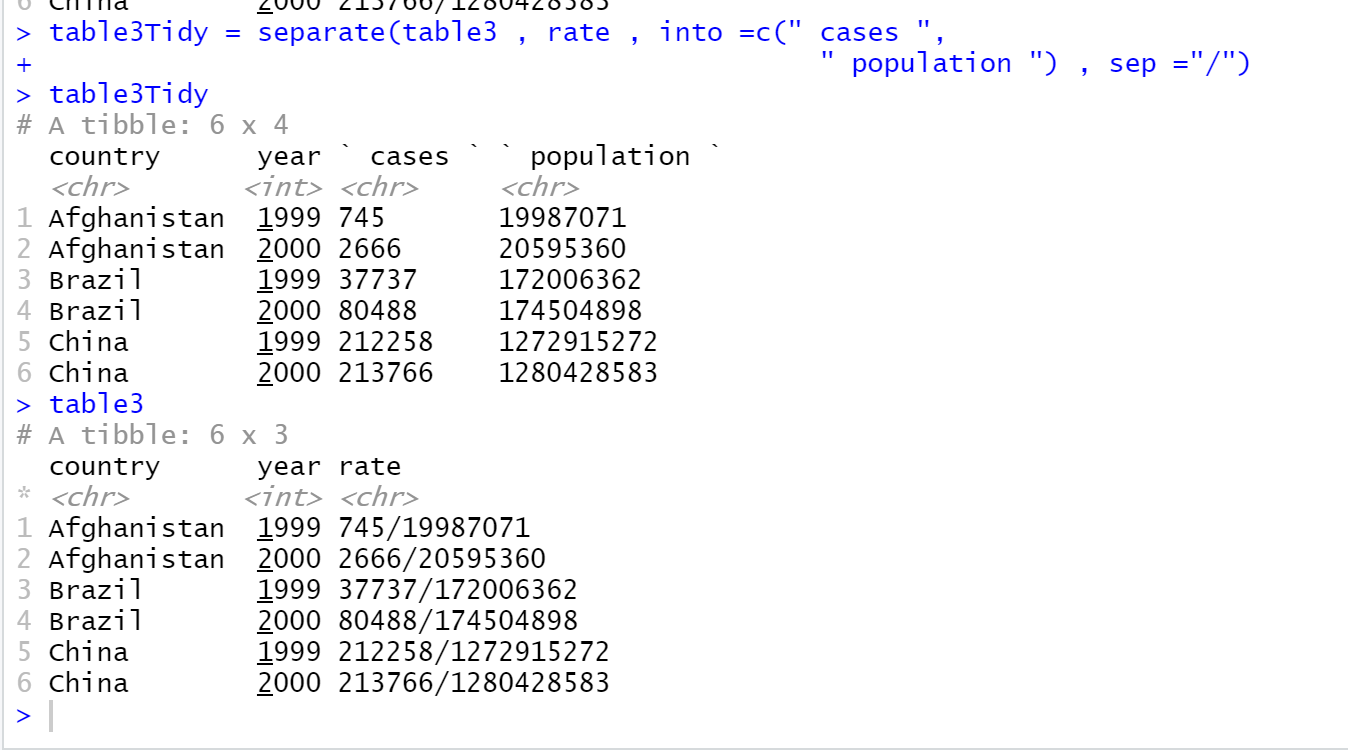
key =c("age", "height", "age", "age", "height"),

value=c(45,186, 50, 37, 156)

)

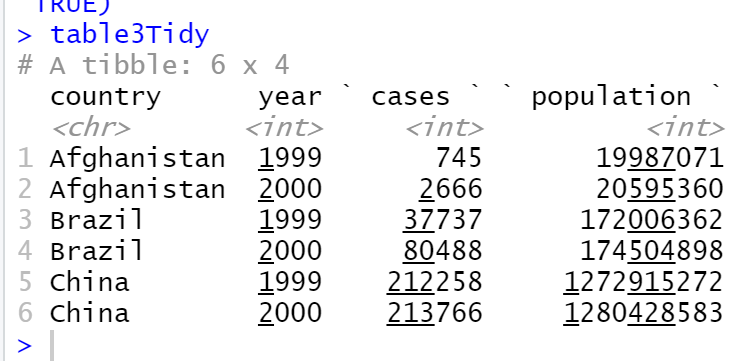
Multiple Variables are stored in both one row and one column

A complicated form of messy data occurs when variables are stored in both one row and one columns. In table3 the column rate contains the two variables cases and population. To fix the problem we use the function **separate().** As arguments, it takes the name of the column to separate, and the names of the columns to separate into. Further, you can specific a character to separate the column; here it is ”/”.



If you look at the column types, you will notice that case and population are character columns. This is the **default behaviour in separate(), it leaves the type of the column as is.** Here, it is not very useful as those are really numbers. We can ask separate() to try and convert to better types using **convert = TRUE**:

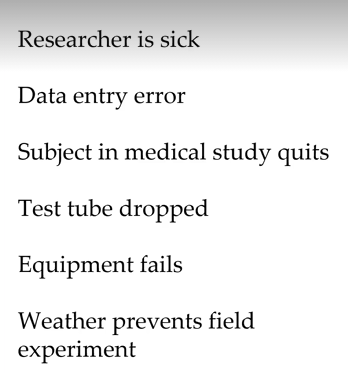
separate ( table3 , rate , into =c(" cases ", " population ") , sep ="/", convert = TRUE )



Here, you have learned about a few important functions from the **tidyverse** package. It contains more functions to tidy datasets. If you would like to learn about them, you can consult the Wickham’s book (available online [11]).

## Missing Data

Two main types of missing data can be distinguished [4]:



• MAR: missing at random. In this case, conclusions based on data with missing values should not differ from conclusions based on complete datasets.

• MNAR: missing not at random is a more serious issue. In this case, it might be wise to check the data gathering process further. The nature of the pattern needs to be understood before one can interpret the results correctly. An example of MNAR is men failing to fill in a depression survey because of their level of depression.

Note that if the missing data pattern is not random, then there is no statistical approach to solve the problem. Thus, techniques for dealing with missing observations assume that the pattern of data loss is **random[7**].

In the following we are working with the IRIS dataset. It can accessed by the name “iris”. First, we will create an iris data frame with 10 % randomly missing values using the prodNA function from the missForest package.

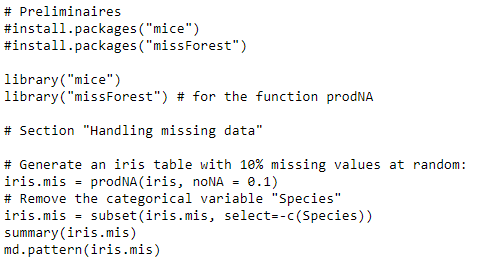


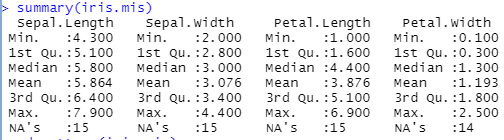
We want to focus on numerical values and remove the categorical variable Species.



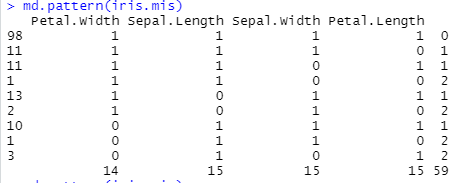
The **mice package provides a nice function md.pattern()** to get an understanding of the pattern of missing data.







Output of md.pattern(iris.mis)



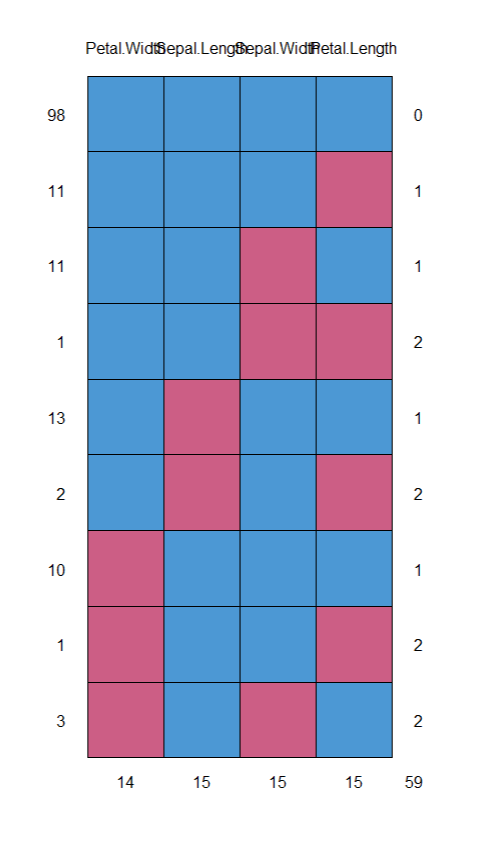


Figure 2: output of iris.mis as pattern. Blue pattern are 1 and red is for zero.

* 1. Deleting data

If the amount of missing data is very small relatively to the size of the dataset, then leaving out the few samples with missing features may be the best strategy in order not to bias the analysis. The rows in iris.mis containing missing values can be removed as follows:

# Section "Deleting data"

iris.del = na.omit(iris.mis)

summary(iris.del)

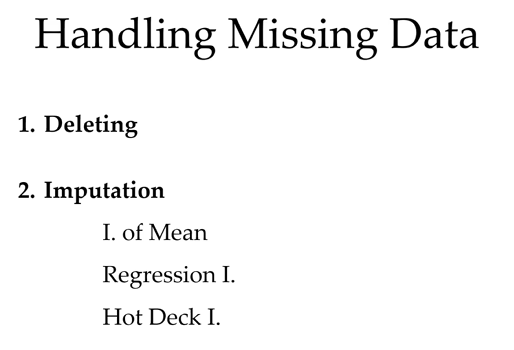
md.pattern(iris.del)

However, leaving out available data points deprives the data of some amount of information. It can substantially lower the sample size, leading to a lack of power. This is especially true if there are many variables involved in the analysis, each with data missing for a few cases. Depending on the situation, you may want to look for other fixes than deleting potentially useful data points from your dataset. Imputation is such a fix.

Imputation of the Mean: One of the simplest imputation methods is to replace the missing observations of a particular variable x by the mean ¯x of the observed values of x.

Two classes of missing data:

**Missing at random and not at random**



**Imputation of the Mean**

Exercise 3.2.2 Apply this approach to the variables Sepal.Length from iris.mis.

Exercise 3.2.3 (Advanced) Write a function that takes a data frame as an argument, performs Imputation of the Mean, and returns a data frame with imputed values.

**Regression Imputation**

With Imputation of the Mean all the missing values are replaced by the same number. This may lead to underestimates of the variability in the data. In contrast, Regression Imputation computes a number for each missing value. The approach works in two steps. Suppose we have the variables x1, x2, . . . xk, and x1 has missing values. 1. Estimate the relationship between x1 and x2, . . . , xk, for instance, by linear regression. 2. Use the regression model to predict the missing values in x1.

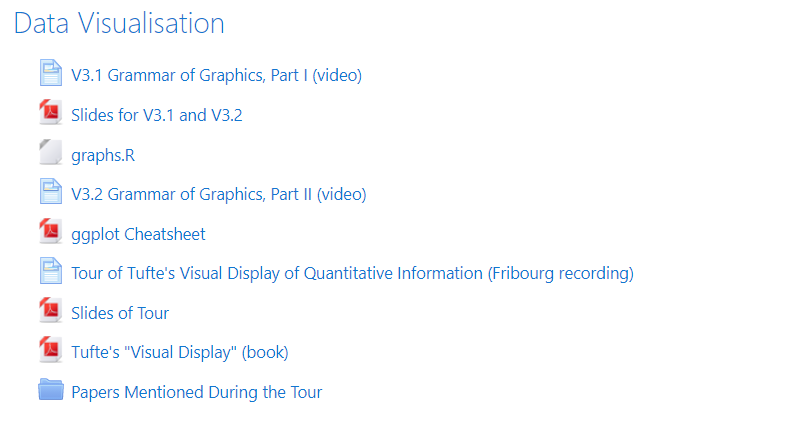
**Hot Deck Imputation: Predictive Mean Matching**

Regression Imputation may lead to predictions that are not valid, for example, negative lengths in the iris data. Hot deck imputation overcomes this problem. In this family of methods, missing values are imputed by copying values from similar entries in the same dataset. For instance, in Random Hot Deck Imputation, a value is chosen randomly, and uniformly. Another Hot Deck method is Predictive Mean Matching (PMM). It is similar to the regression method (from the previous section), but has two additional steps. Suppose we have, as above, the variables x1, x2, . . . xk, and x1 has missing values. 1. Estimate the relationship between x1 and x2, . . . , xk by linear regression. 2. Use the regression model to make predictions for all the values in x1, for the missing ones and for the non-missing ones. 3. Suppose there is a missing value in the i-th row, i.e. x1i is missing. An let us denote the corresponding predicted value from step 2 with ˆx1i . Regression imputation would substitute the missing value by ˆx1i . In contrast, PMM finds a set of rows with non-missing values, whose predicted values are close to the predicted value ˆx1i . 4. From among those close cases, one case is randomly chosen and its observed value is used to substitute for the missing value x1i .

1. Data Visualisation

4.1 Introduction to ggplot

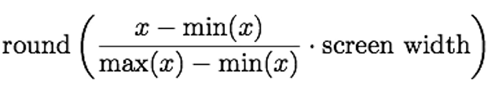
R has **built-in functions** for **visualisation**, and there exist a number of packages for the task. ggplot is one of the most popular ones, or maybe the most popular. It is very flexible and has a logic for the mapping between the data and its representation, allowing a graphical construction of plots. The author of the package is Hadley Wickham; consequently, ggplot assumes data in tidy format (see Section 3.1).

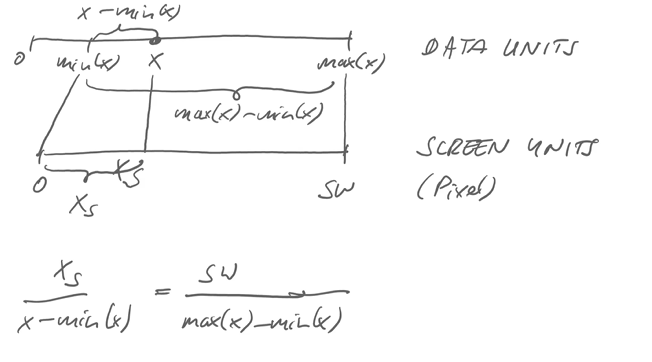


Grammar of graphics: part 1:

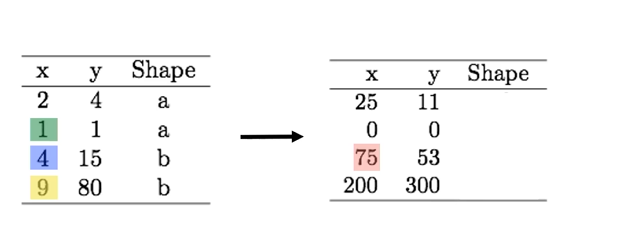
Main components of grammar:

1. Data, scatter plot – geometric plot
2. Geometric object
3. Aestetic mappings - Scales: covert numbers in data to units for display (pixels)



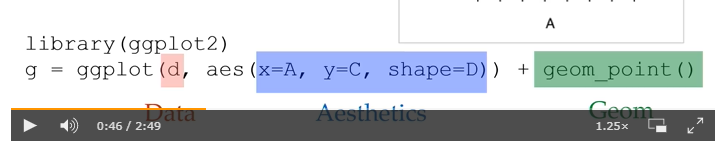




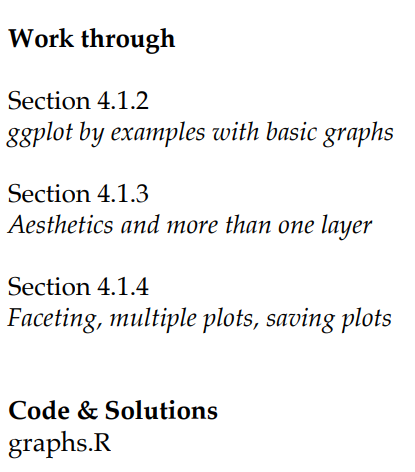


Grammar of graphics – part 2:

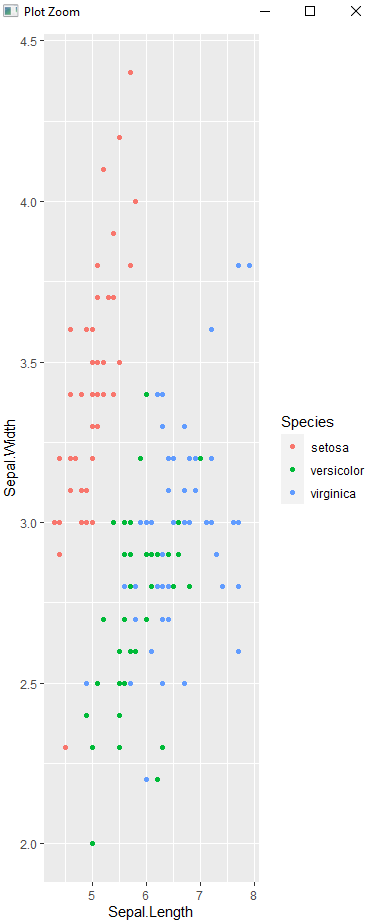
1. We take the data and put it in a data frame
2. Than we load the ggplot library
3. Next we define our three elements:data, geometric and aesthetics



Multiple layers:



Exercise 4.1.2: ggplot by examples with basic graphs:



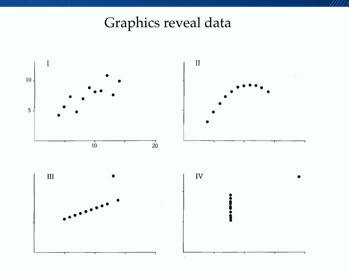
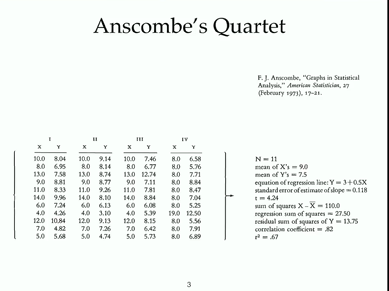
Exercise 4.1.1 In this exercise you will use the built-in data set mtcars. You can learn about the dataset with R’s help function: help(mtcars). Make a scatterplot as above. Assign the variable mpg to the x-axis, wt to the y-axis, and gear to the colour. This does not lead to the desired result. Why? (Compare the data type of Species and gear). How can you solve the problem?

Motor Trend Car Road Tests

## **Tour of Tufte's Visual Display of Quantitative Information**

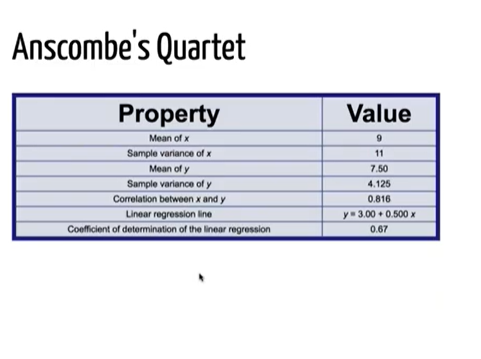
The visual display of quantitative Information

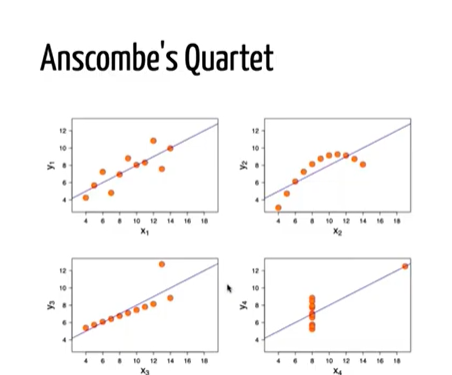
Anscombe’s Quartet:



Famous examples:

In statistics we could say, the numbers are the same! However, if we look at the visualisation: we see quite smth difference





Tufte says: reviewing and displaying the data are different.

2 main themes: 1. Tell the truth -graphical integrety

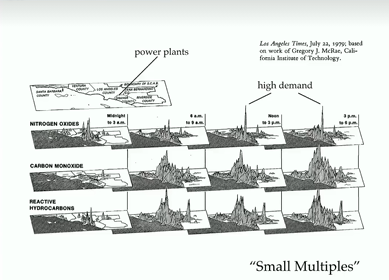
1. do it effectively with clarity and precision – design aesthetics

how do you communicate with your data?

Examples:

1. y axis- various stop during the path paris – lyon. On the x axis you have the whole day.
2. What do you see now?: you see all the trains going- how long the stop.. you see crossings.
3. What he likes about the plot: you can relieve all information in one plot.

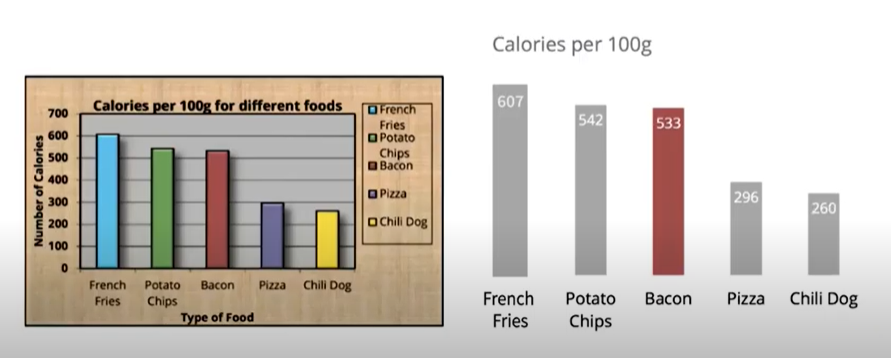
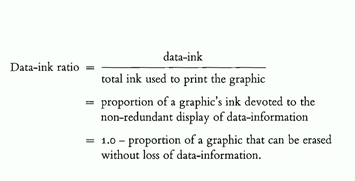
Another plot

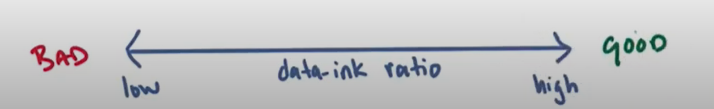
you map different aspects in one plot. What it shows 3 diffent kind of pollution. Diff. times on the day. And a little geographic maps.

Principles of Graphical Excellence:

Give the viewer

1. The greatest number of ideas
2. In the shortest time
3. With the least ink in the smallest space





X is not a data – it gives a frame

He suggest: above all else show the data

Maximize the data-ink ratio, within reason

-erase non-data-ink, within reason

-erase redundant data-ink, within reason

**What is visualisation: visual explaination- help show the relationship between topics.**