Data Literacy with R for Students of Humanities

Silvie Cinková

2025-07-18

Table of contents

1 Welcome

- This course should stretch your frustration tolerance.
  - You will be lost without everyday self-study.
  - If not, maybe you were too modest in your self-assessment.
- No question is stupid. *Please repeat* is excellent!
  - You are asking for the shy ones.
  - Teacher gets easily wrong or incoherent, like most people.
- Hazard **loud guesses**.
  - Often legit assumptions stumble on system-inherent idiosyncrasies.
  - You are hardly alone thinking so.

2 Welcome

RStudio on cloud

https://aic.ufal.mff.cuni.cz/jlab/

Learning materials on github

https://github.com/ufal/R_BEGINNERS_SHORT

This course in the course catalogue

https://is.cuni.cz/studium/eng/predmety/index.php?do=predmet&kod=NPFL146

DataCamp

https://www.datacamp.com/users/sign_in

3 R vs. Python for Data Science

- Which is preferred in your research domain?
  - NLP, DH: both
- R probably best at tabular data (manipulation, plotting)
- Nice comparison at statology.org:

https://www.statology.org/r-vs-python-for-data-science-a-comprehensive-comparison/

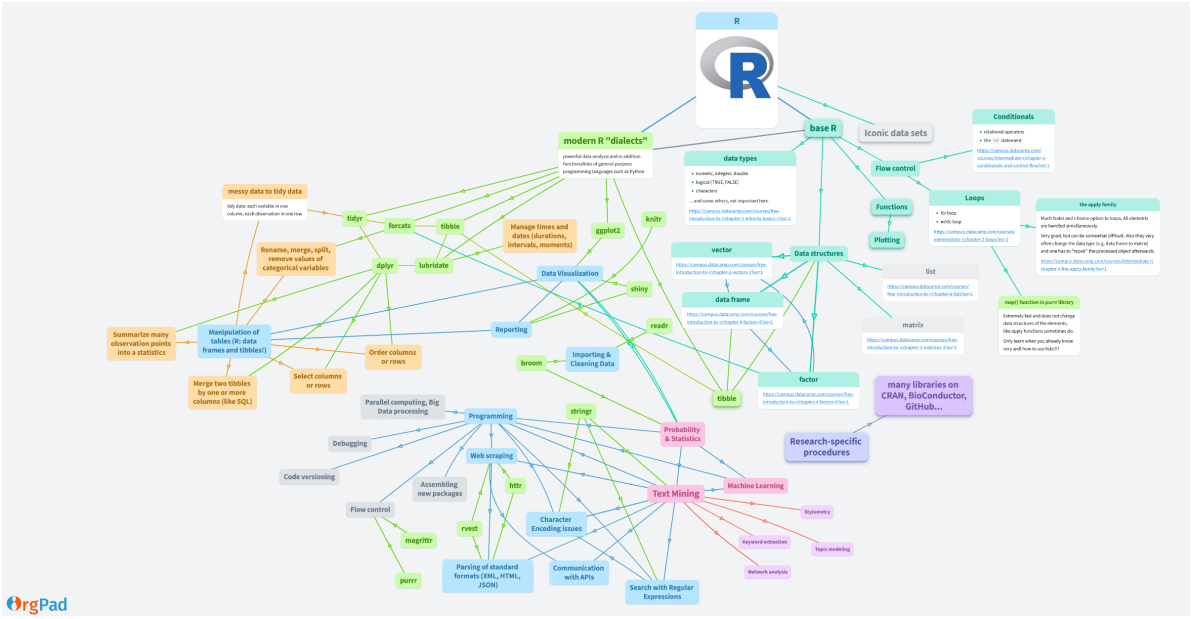
R used to be a highly specialized language for statistical computing unlike general programming languages such as Python. Its main advantage was vectorization: the ability to perform the same operation on all elements (e.g. table rows) simultaneously. Computer processors in 1990s could normally not do it, the general-purpose languages had to process one value after another (a *loop*, present in possibly all programming languages). With modern processors, general-purpose languages can vectorize many operations and R accrued functionalities of a general-purpose language, so the differences are not so sharp any more. R possibly (still) better at data science. For application programming, use Python. Most important: use what your scientific community uses and supports with libraries.

4 Mind map of R

- Core concepts, links to DataCamp classes https://orgpad.info/s/SKpWCNY4P7w

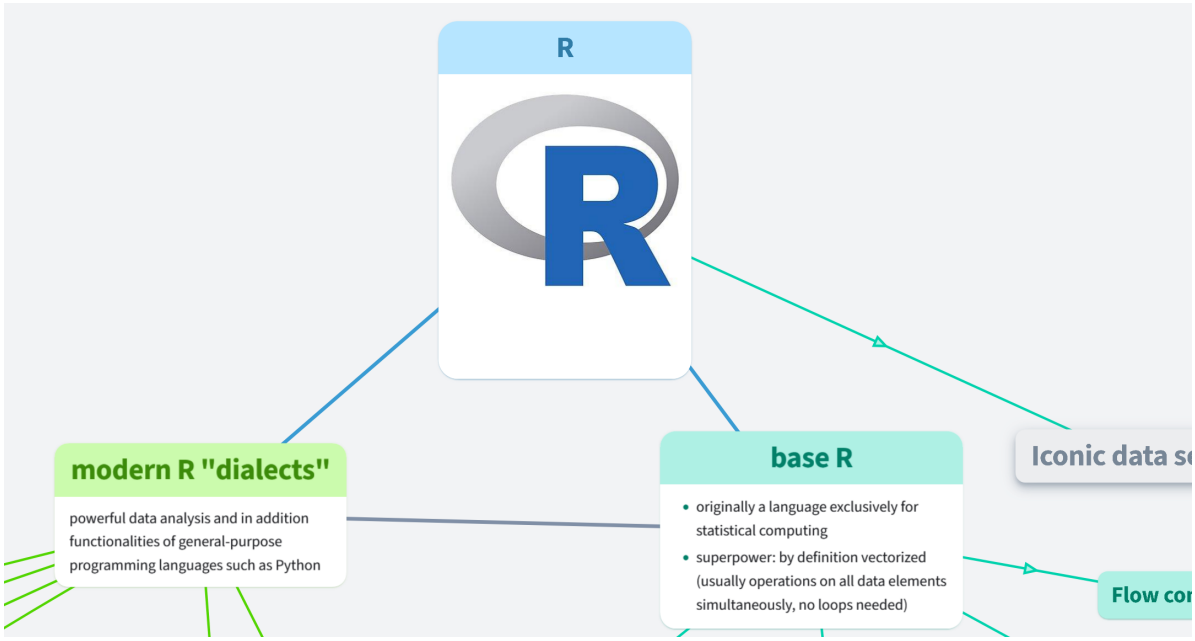
I will walk you through my mind map of R functionalities. The QR code takes you where it lives, but snapshots with comments follow on these slides and that should do.

5 The Big Picture



Some cells have content and can be opened on clicking on the live website. Cells of the same colors tend to link libraries or functionalities that belong together.

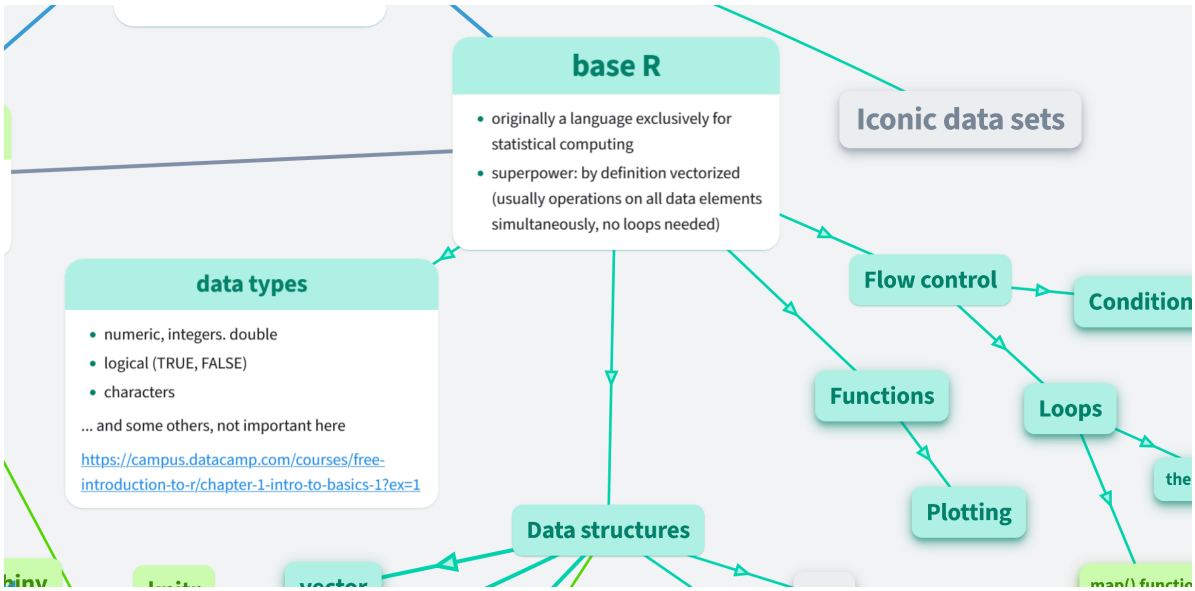
6 Base R vs. “dialects”



Base R: the original core. Great statistical computation work when you had a table to read in; preparation (data wrangling) not so comfortable. We will learn that bit of base R that is useful for data wrangling, with which we are going to spend virtually all time.

Modern “dialects” introduced new elements of syntax. We will learn to use such a family of software libraries for R called `tidyverse`. They are not the only but very influential for data manipulation and plotting. We will learn just the modern plotting dialect called `ggplot2`.

7 Base R



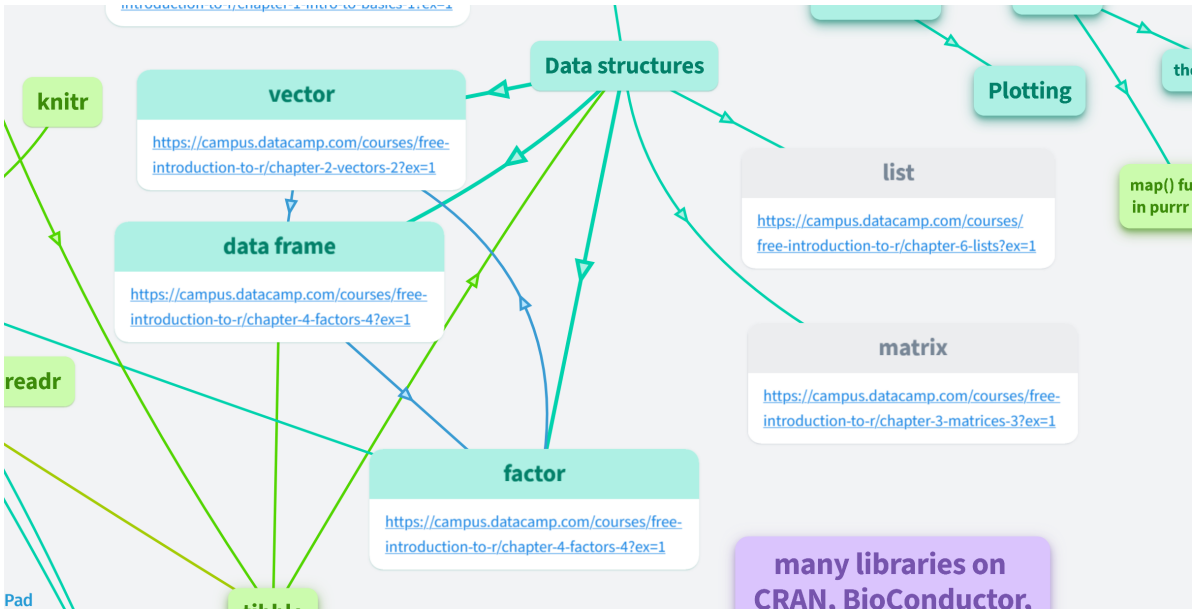
R is optimized for fast operations on columns in tables, and hence it tends to treat all inputs as potential columns of a potential table.

Data types (classes): The observations can be either numbers or strings, or Boolean values (TRUE/FALSE). You can gather the observations into diverse **Data structures** (e.g. tables). You can look at them as nouns in a natural language.

The verbs would be **Functions**. You typically give them some values on input and they return some values on their output. Apart from functions, there are a few control structures: Condition and several types of loops.

"Programming in R" usually refers to base R. Otherwise you "only" code using libraries.

8 Data structures



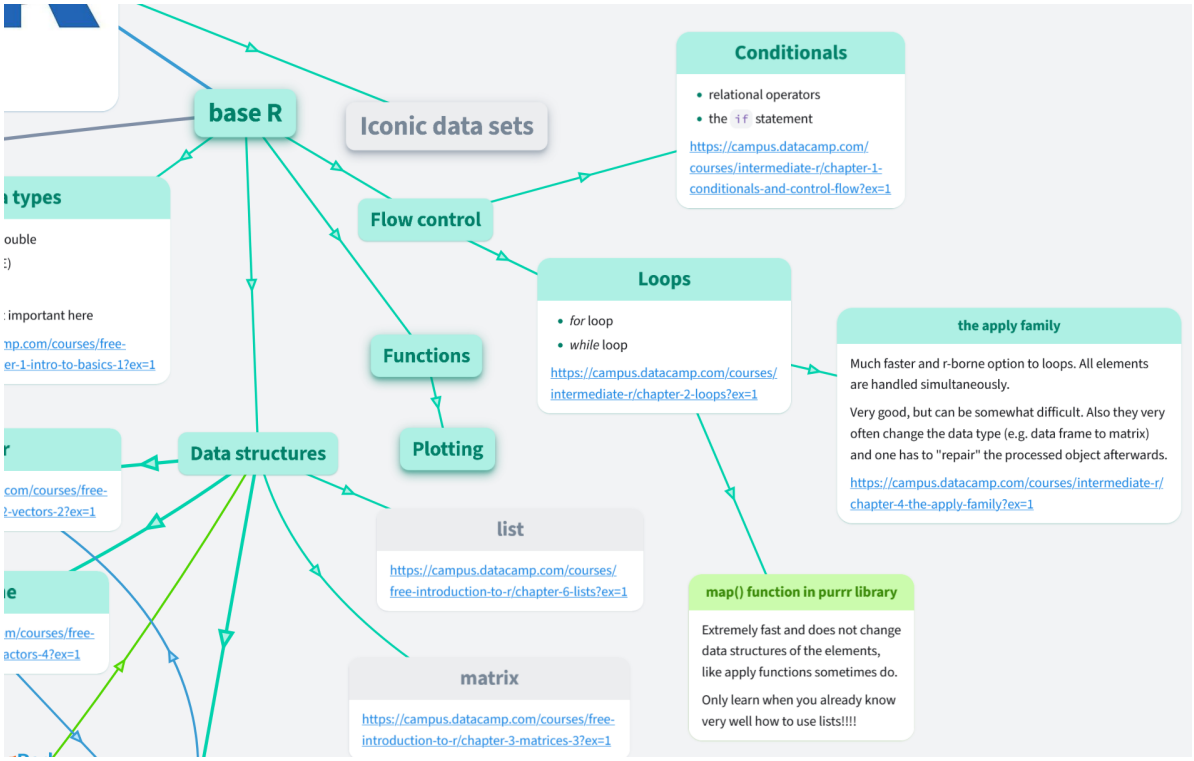
The signature data structure of R is the **vector**: ordered sequence of observations (i.e. position matters).

Each observation = vector element. A single observation = a vector with one element. **Factor** = a vector of character strings that has been told that the strings are values of a categorical statistical variable, so it immediately records for R the frequency of each value, without calling a counting function every time.

When vectors and factors are equally long, they can be adopted as columns into a **data frame**. These are the structures you are going to need regularly. Other important structure is *list*. It can store any other structures as its elements. For instance, you can have a list of several data frames, vectors, and other lists with anything inside.

Matrix is not much relevant for this course. It is actually a vector but it has an additional dimension of columns, which makes it similar to a data frame. However, all matrix columns must be the same data type, whereas in data frame, each column in a data frame can be a different data type (strings, numbers, Boolean values).

9 Functions and programming structures



A **function** is like a machine with a given purpose. Imagine a microwave oven. All it does is heating up, but the results of heating up depend on what you input and what time and intensity you select: Cold water -> boiling water, corn -> popcorn, … You get rubbish when you put in something unintended or select wrong intensity or time. Unlike a microwave, you cannot destroy a function though.

Control structure **Loop**. You can call a function (or a larger piece of code) on a series of input objects, and the code will process one after another.

Control structure **Condition**: It says: "If this condition is met, run this piece of code". For instance, your piece of code takes a number and divides it by two. When you feed it a string instead of a number, it throws and error and stops your script. With a condition saying "do this only to numbers", you make sure that an accidental string would not crash your script.

While loop and conditions are programming universals, base R has a specific family of functions to avoid loops. They are have `apply` in their name. You will see them a lot in base R code. They have a sligthly different syntax than regular functions, so you have to learn about them separately. When you are going to work mainly on data frames (tables), you would probably benefit from learning to use the corresponding functions from the `purrr` library, which is a "dialect" of R.

There is one more notable cell in the image: Iconic data sets. All libraries come with some documentation of their functions, and most demonstrate them on me on data sets that R makes available to all users - built-in data sets. Each new library can add its own built-in data sets. Most R users are familiar with quite a few such built-in data sets. These data sets come from diverse domains from botany to flight schedules.

10 Tidyverse

https://www.tidyverse.org/

R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

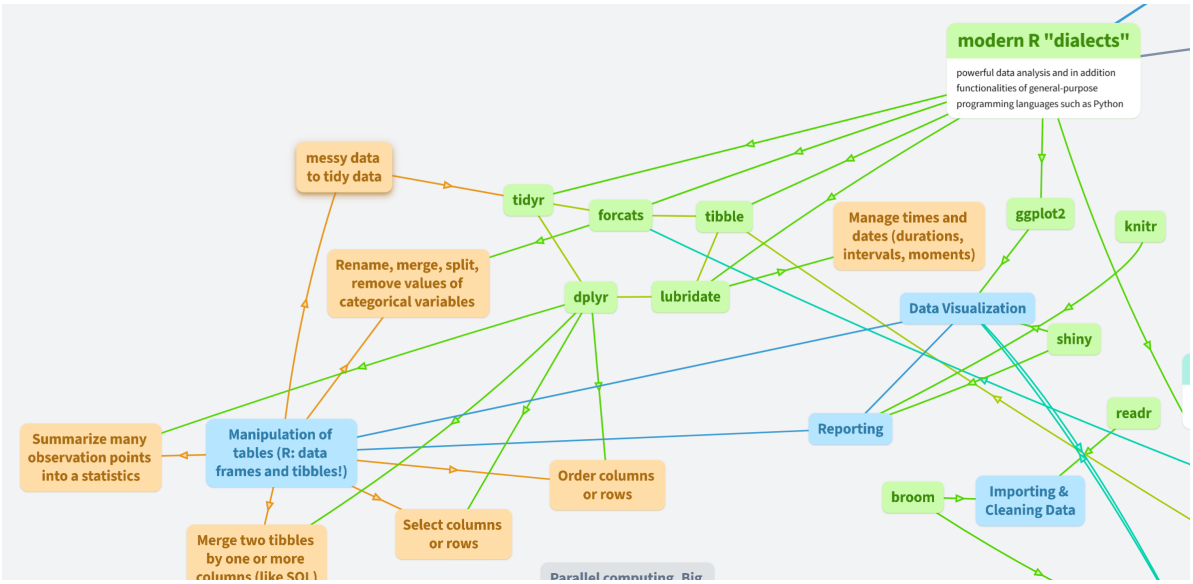
```
install.packages("tidyverse")
```

In our course, we are going to focus on one modern R "dialect" for data manipulation, the `tidyverse`. It is a family of software libraries that has a fabulous library for plotting, too. Here you see icons of the `tidyverse` libraries. The `tidyverse` is associated with the name *Hadley Wickham*, the initial author, but it became so popular that it has grown to an entire ecosystem of complementary libraries with many authors and contributors. The core idea of `tidyverse` is `tidy data`, which means each table row represents one observation and each column represents one variable, and once you stick to this data philosophy, you can use an incredibly user-friendly and efficient toolkit.

So when you work with data frames, you mostly need to filter away some rows or select just some columns, or to add another column based on a computation on other columns or something else. Or you have the data scattered in several different tables and want to merge them into one, just as if you query a relational database. And this is what the `dplyr` library is for. `tidyr` is a library that helps you make your data comply with the `tidy` principles. `stringr` makes it easy for you to operate on strings (text search, replace, switch case etc.). `readr` is for reading files into R objects and saving R objects to files, e.g. a data frame to a `.csv` file. `lubridate` is for dates and times. Without it, they are just strings. When you need e.g. to compute duration from two dates and times, call `lubridate`. `ggplot2` is the plotting library.

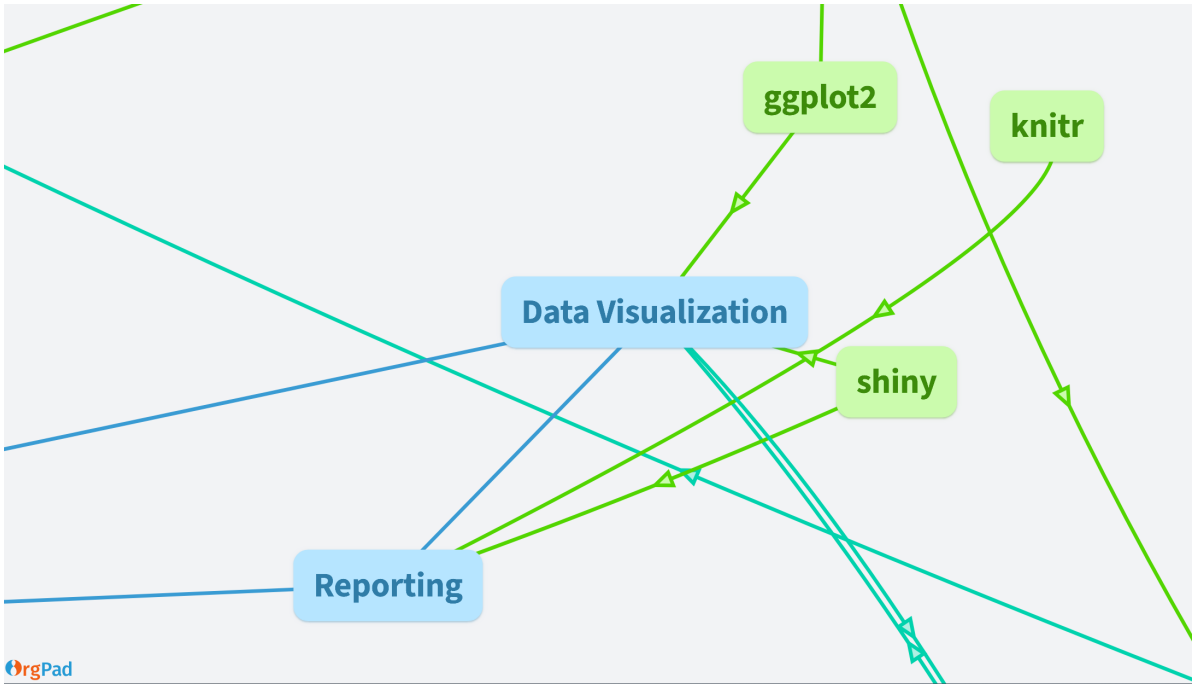
The last three libraries are not so obvious at this moment, but for the sake of completeness: `purrr` processes things in lists. `forcats` works with factors (those vectors that know that they are categorical variables and count frequencies of their values). `tibble` adjusts base R data frames to fit better with `tidyverse` packages.

11 R for Data Science



The purpose of Data Science is modeling of relations between statistical variables, for instance people's weight and people's height. We will concentrate on how to prepare your data to give just the variables you want, ready for visualization or advanced statistical modeling. The main tasks are represented by the blue cells: **Importing**, **Manipulation** (aka **wrangling**, **massaging**), and **Visualization**, mostly with the `tidyverse` libraries. You will also learn how to present your code, plots and texts in a single document that you can render as a PDF, MS Word or html document. That's the **Reporting** cell.

12 Reporting



I will first of all mention *Reporting*. It is admittedly not the most important programming skill, but it can help you keep great notes alongside your code right from the start of the course, so you can re-run the scripts and edit the comments any time.

There are interactive reporting tools in RStudio, which are based on a library called `knitr` (outside `tidyverse`). You write your text like plain text with a set of simple formatting tags. This format is called `Markdown` and you may know it from note-taking apps such as Notion or Obsidian. R uses a slightly modified "flavor" of Markdown called *RMarkdown*, in which you can combine executable code and free text. You can even insert images and other external objects.

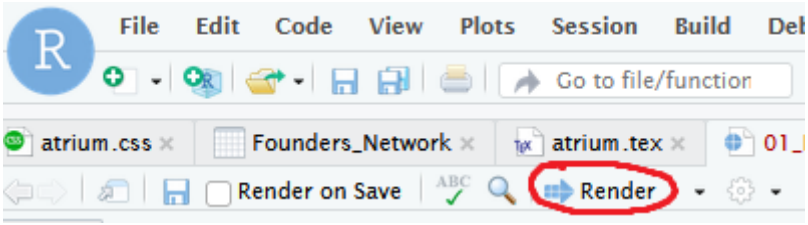
RStudio will offer you three reporting tools: `RMarkdown`, `RNotebook`, and `Quarto`. All of them generate nice reports and slides using markdown and `knitr`. All are good and give a similar user experience, but Quarto is newest and likely to replace the older ones in RStudio, so it is probably wise to stick to Quarto.

Just on the margin: reporting is more than reports you can print or run on your computer. Other popular formats are interactive websites and dashboards, but these are beyond our current scope.

13 Note keeping tip

How to keep notes in RStudio

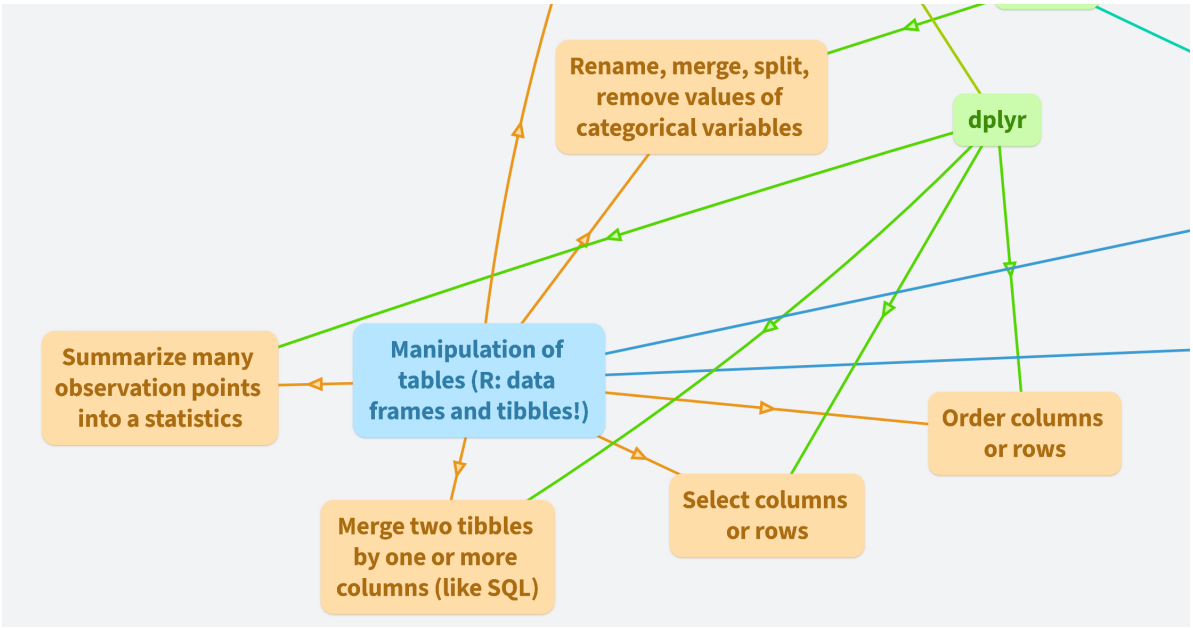
- Source file of this presentation: `01_Introduction.qmd`.
- Edit your local copy to add your notes in Markdown.
- To preserve the slides structure, write your notes inside `::: notes :::`
- You can render it as pdf/html even if you ruin the slides rendering.
- How to keep the presentation structure
  - https://quarto.org/docs/presentations/



To keep notes in RStudio, you open a `.qmd` or `.Rmd` file. I write my teaching materials in Quarto and my default output is `revealjs` slides. I write the accompanying text into a field called speaker's notes. It is not a Markdown but a `revealjs` functionality. They are hidden to the audience during a slide show, but you see them when I render the same content in html or pdf, including the tags. A block of notes like this one must always have one empty line (with its own number) before and one after, otherwise the notes appear in your slide along with the tag.

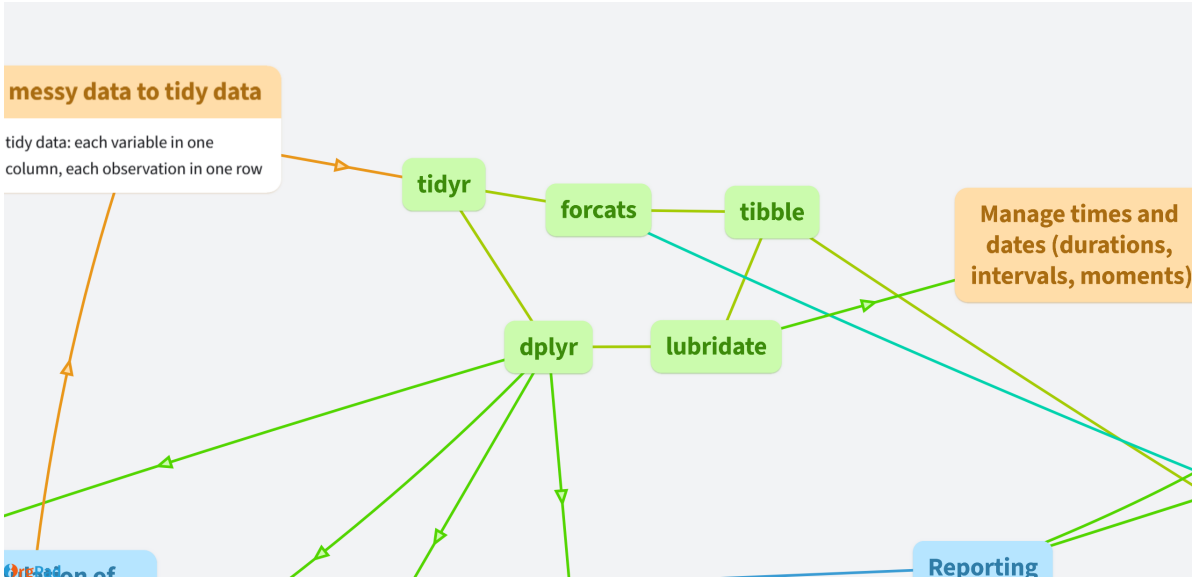
You can also include latex commands and html code into markdown documents for better styling, but we will not learn this in this course. The goal here is that you have reproducible notes, and that you will get with the markdown template you see when you open a new markdown file. Here is the user guide to Quarto: https://quarto.org/

14 Data manipulation - the core skill



The `dplyr` library is a big library to arrange rows according to the value of one or several variables, delete unwanted columns or rows, add new observations, or make a summary of your observations, for instance count the average of a variable or frequencies of the values of a categorical variable. With `dplyr`, you can also combine selected columns from several tables into a new table. Then you can visualize variables in a plot with `ggplot2`, which in the mind map is linked to a blue cell inscribed *Data Visualization*. `dplyr` and `ggplot2` are our main tools. When you need to split or merge columns (or to restructure your data frame more thoroughly), you use `tidyr`.

15 Data frame wrangling

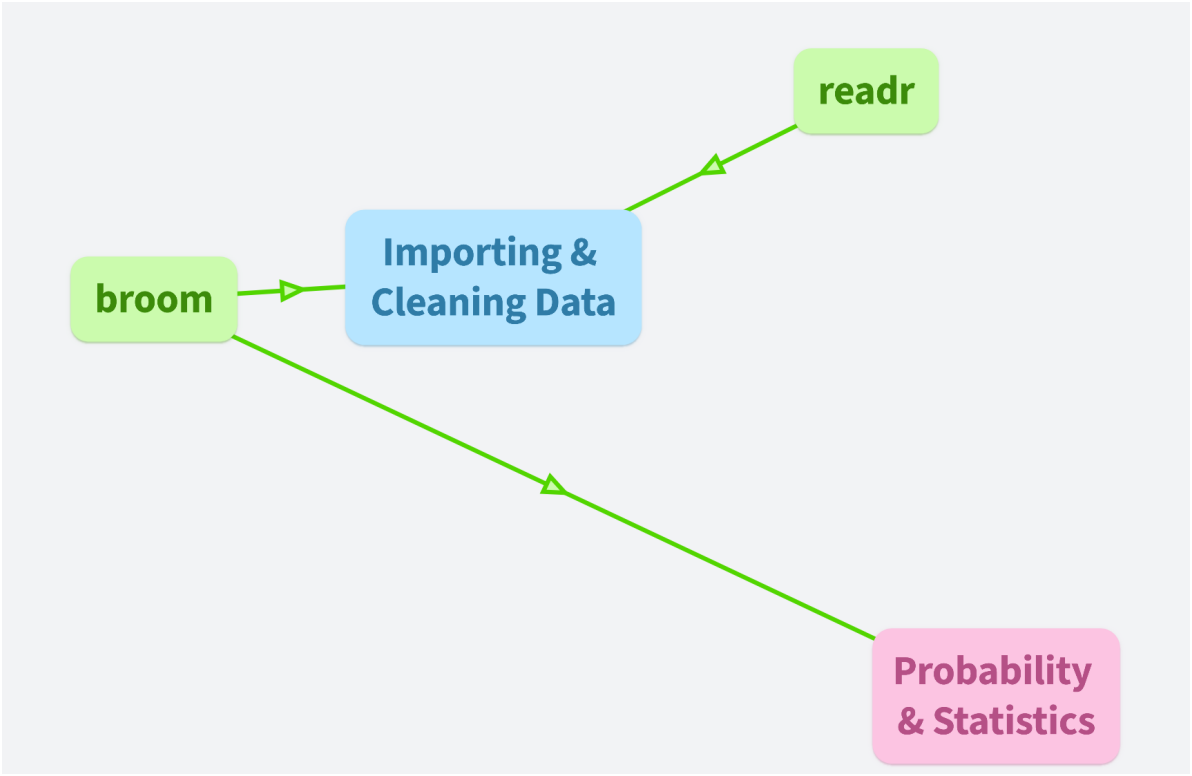


A table is called **data frame** in base R. In `tidyverse`, you will often meet the term `tibble`. A `tibble` is a data frame adjusted to the tidyverse packages. You do not have to worry about the difference. The `tidyverse` functions convert an input data frame to `tibble` behind the scenes.

The `lubridate` library makes dates and times interpretable. Without it, they are just character strings.

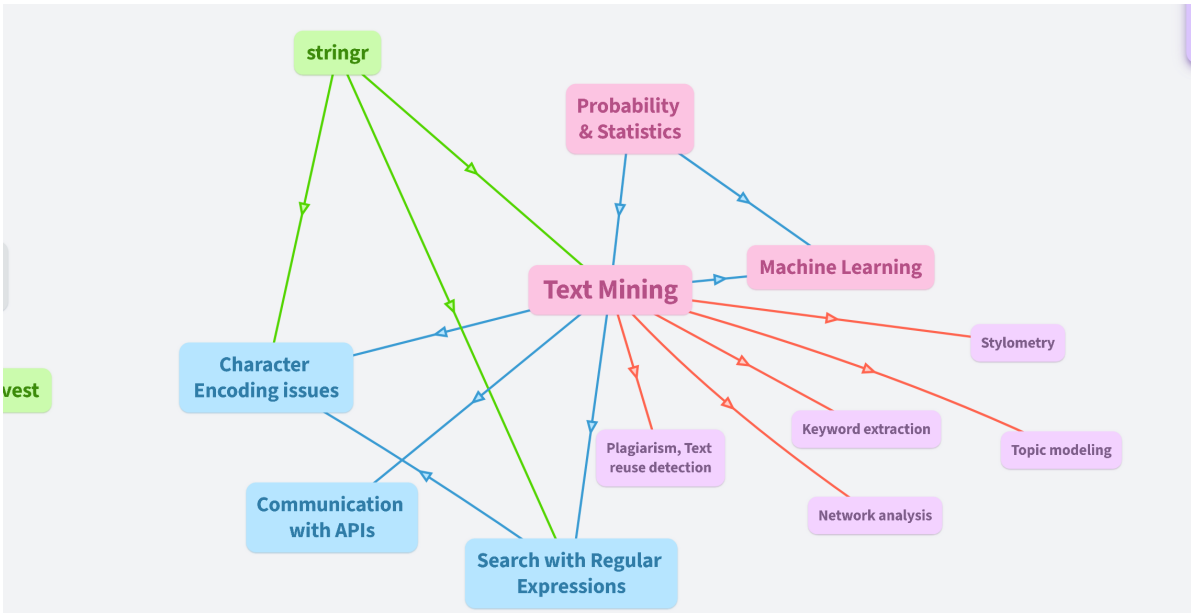
`forcats` is a library that makes it easier to process categorical variables in data frames.

16 Data import: `readr`



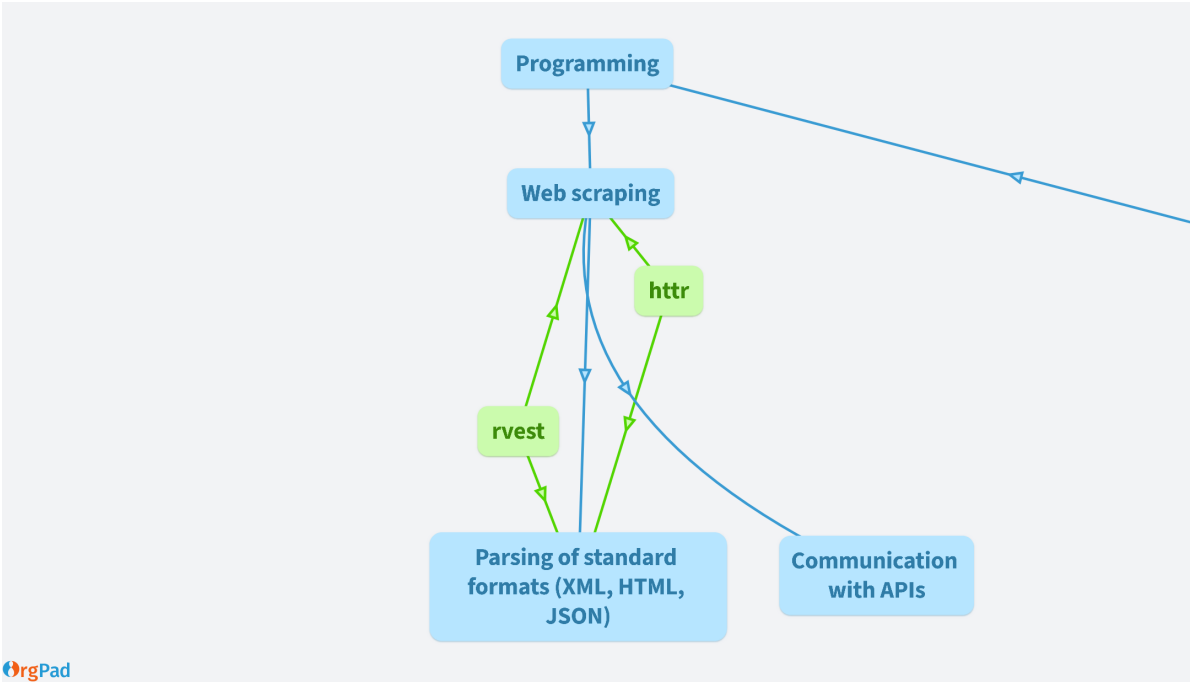
Here you see two libraries to import data. We will work with `readr` and its friends outside the `tidyverse` to read files into R objects (mostly data frames). You would like to use `broom` if you analyzed regression models, but that's beyond our scope.

17 Processing text (strings)



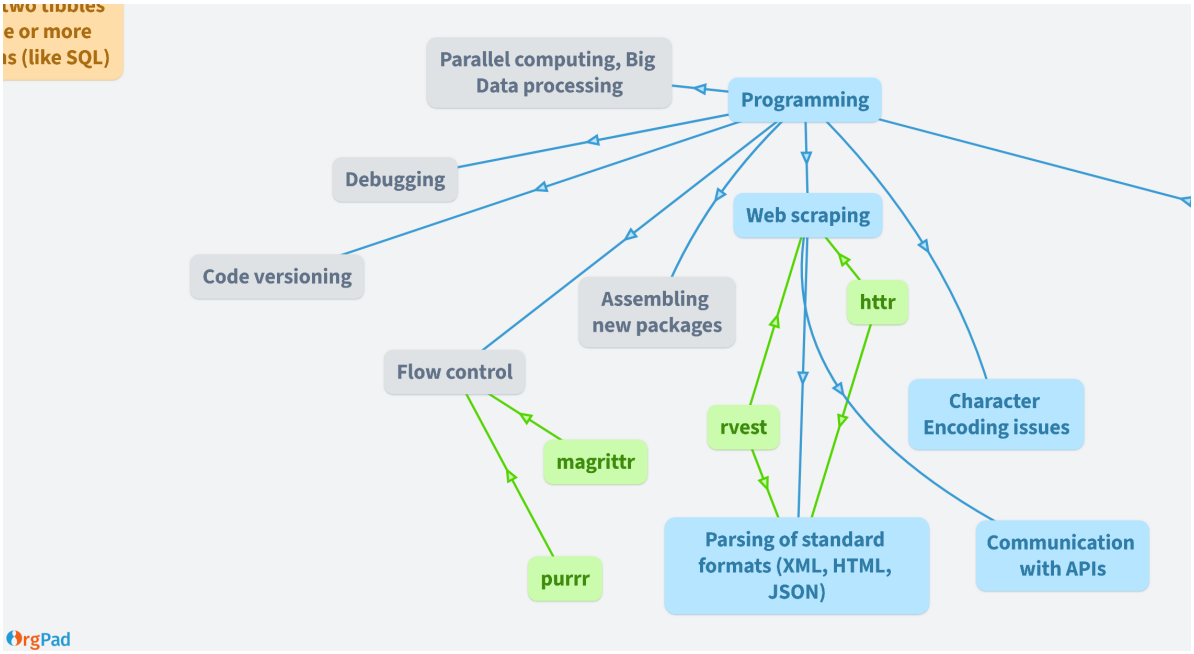
Here I want to spotlight `stringr`- the `tidyverse` library that operates on strings. When you deal with data frames, you often want to split columns based on some textual pattern in the values; for instance, split names and surnames in two columns on the space between them. `stringr` allows you to search for more complex patterns with so-called **Regular Expressions**. But it is useful in much more advanced use cases, for instance as a support of specialized text mining libraries.

18 Extracting external data



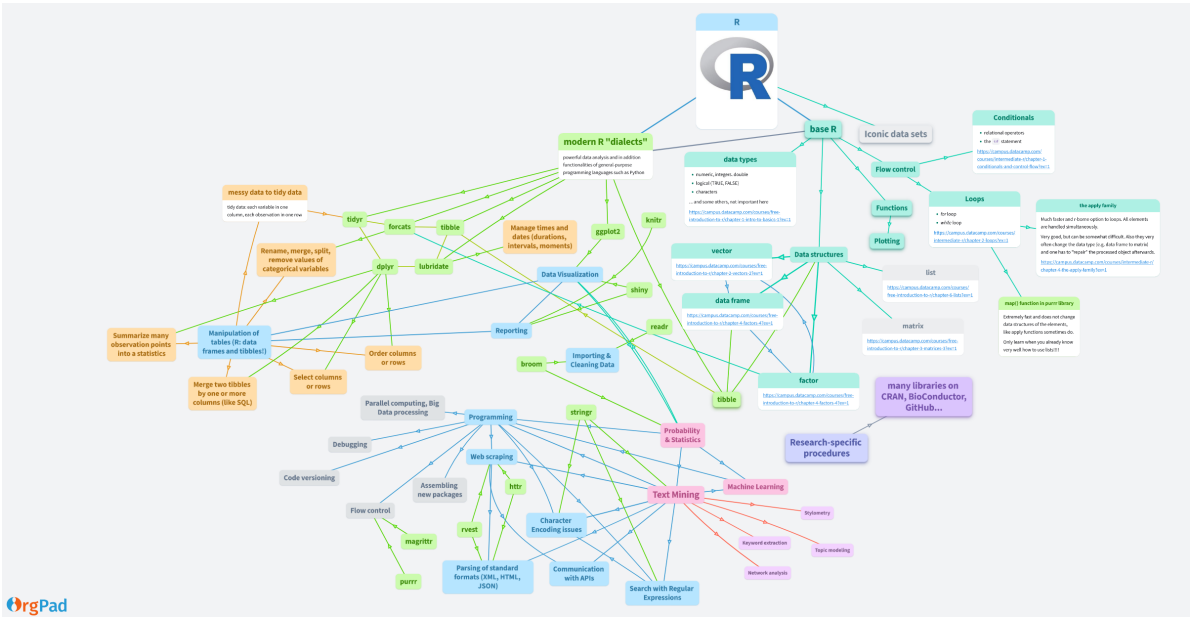
This course will pretend that your initial input is always a built-in data set or a CSV file, but in the real life this is often not the case. Typically, you download the data from an external web service that communicates with you over an API (Automated Programming Interface) and returns the required data in JSON, which is not a tabular format. We will peek on this area as well, without aspiring for true mastership. `tidyverse` helps with two libraries `rvest` and `httr2` (`httr` in the image is obsolete).

19 Programming



With R you can do a lot of programming beyond data manipulation and analysis, even with data so big that you need several computers working on them in parallel. This is beyond our scope, but we will at least take a look at the powerful library `purrr` that allows you to process many data frames at the same time.

20 That's R!


R
base R
modern R "dialects"
powerful data analysis and in addition functionalities of general-purpose programming languages such as Python
Iconic data sets
Conditionals
data types
numeric, integers, double
logical (TRUE, FALSE)
characters
... and some others, not important here
Flow control
Functions
Loops
Plotting
messy data to tidy data
tidy data: each variable in one column, each observation in one row
tidyr
forcats
tibble
Manage times and dates (durations, intervals, moments)
ggplot2
knitr
Rename, merge, split, remove values of categorical variables
dplyr
lubridate
Data Visualization
shiny
vector
Data structures
data frame
list
matrix
factor
Summarize many observation points into a statistics
Manipulation of tables (R: data frames and tibbles!)
Reporting
readr
Order columns or rows
Select columns or rows
Merge two tibbles by one or more columns (like SQL)
broom
Importing & Cleaning Data
tibble
many libraries on CRAN, BioConductor, GitHub...
Parallel computing, Big Data processing
Programming
stringr
Probability & Statistics
Research-specific procedures
Debugging
Web scraping
Code versioning
Assembling new packages
httr
Machine Learning
Text Mining
Flow control
rvest
Character Encoding issues
magrittr
Stylometry
Keyword extraction
Topic modeling
Network analysis
purrr
Parsing of standard formats (XML, HTML, JSON)
Communication with APIs
Search with Regular Expressions
OrgPad