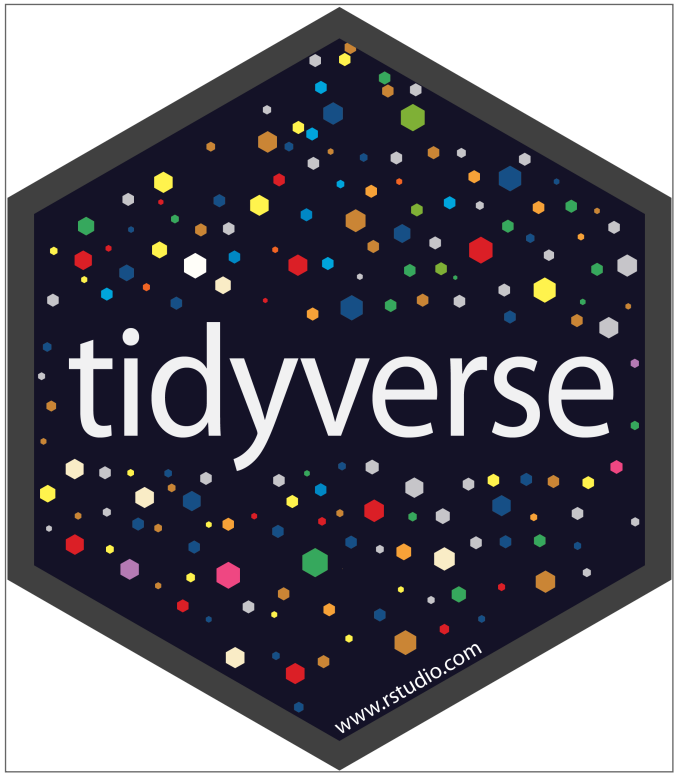
R Environment:

open source platform for data processing: case studies

Krzysztof Banas
19.10.2022

Overview

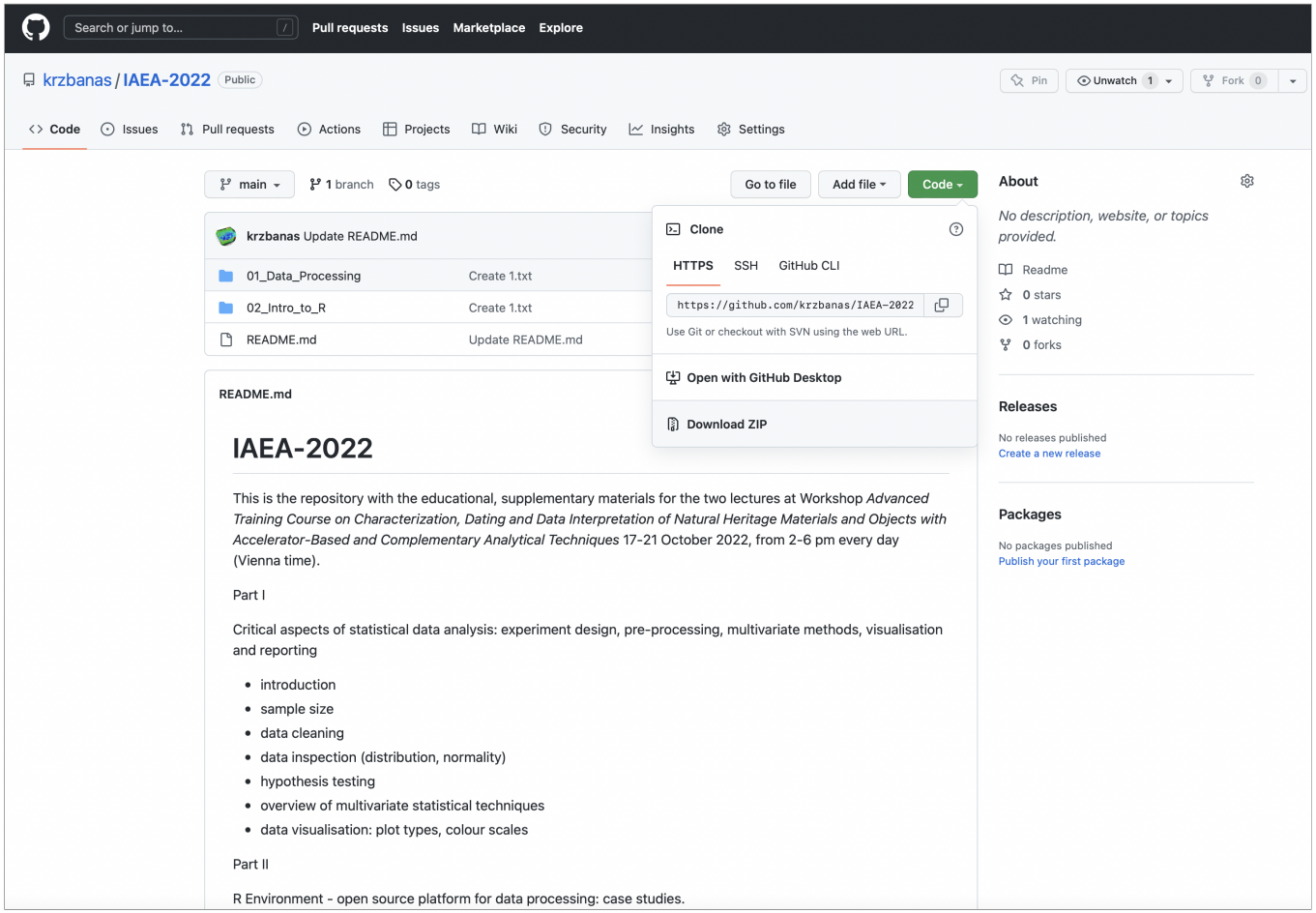
- Intro 👋
- Workshop materials ⬇️
- Time 🕘
- Lecture 3.2 plan 📋



Lecture 3.2 plan

1. What is R? How can it ease the burden of repeated reporting?
2. Basic functions for manipulating data
3. Using R effectively
4. More data manipulation
5. Visualizing data
6. A peek at advanced topics

Accessing workshop materials



https://github.com/krzbanas/IAEA-2022
click big green Code button and select “Download ZIP”

What is R?

R is an open-source (**free**!) scripting language for working with data

imported data shows up here

The benefits of R

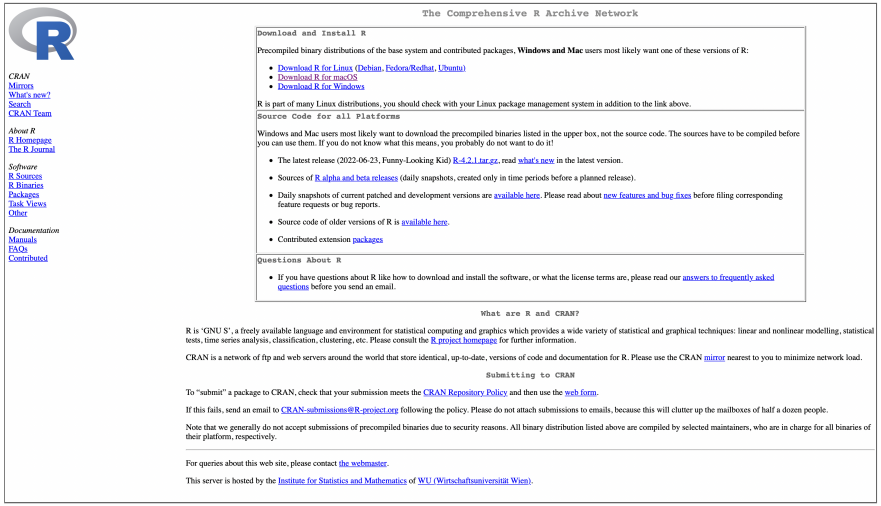
The magic of R is that it's **reproducible** (by someone else *or* by yourself in one year)

Data is separated from code

code can also
go here

Download and install R

- **https://cran.r-project.org**
- open-source
- multi OS support
- Windows
- MacOS
- Linux (Debian, Ubuntu, Fedora/Redhat)



The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2022-06-23, Funny-Looking Kid) R-4.2.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

If this fails, send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

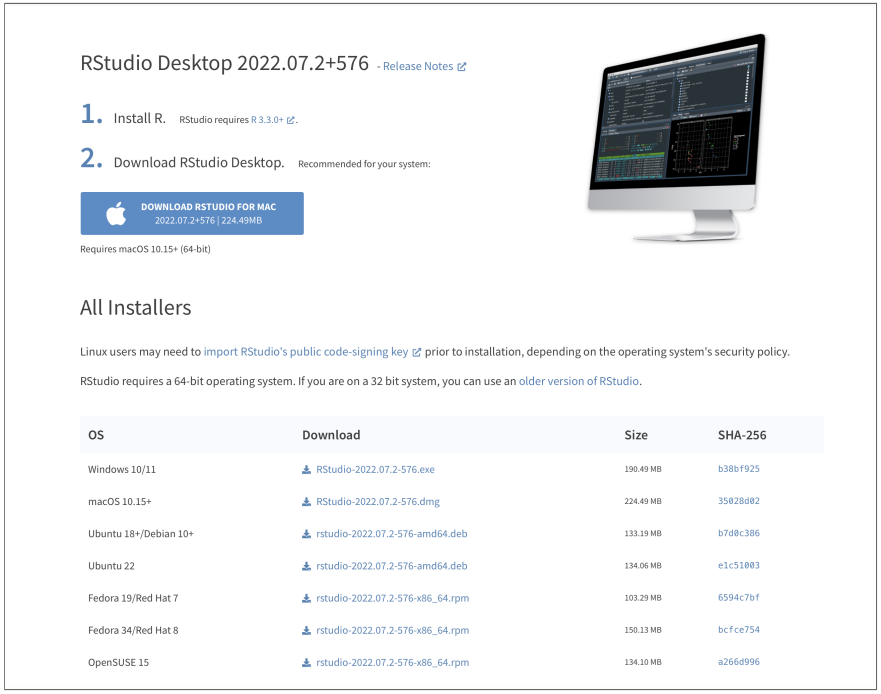
Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all binaries of their platform, respectively.

For queries about this web site, please contact the webmaster.

This server is hosted by the Institute for Statistics and Mathematics of WU (Wirtschaftsuniversität Wien).

Download and install RStudio (Posit in the future)

- **https://www.rstudio.com**
- open-source
- multi OS support
- Windows
- MacOS code *can* go here
- Linux (Debian, Ubuntu, Fedora/Redhat)



RStudio Desktop 2022.07.2+576 - Release Notes

1. Install R. RStudio requires R 3.3.0+.
2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR MAC
2022.07.2+576 | 224.49MB

Requires macOS 10.15+ (64-bit)

All Installers

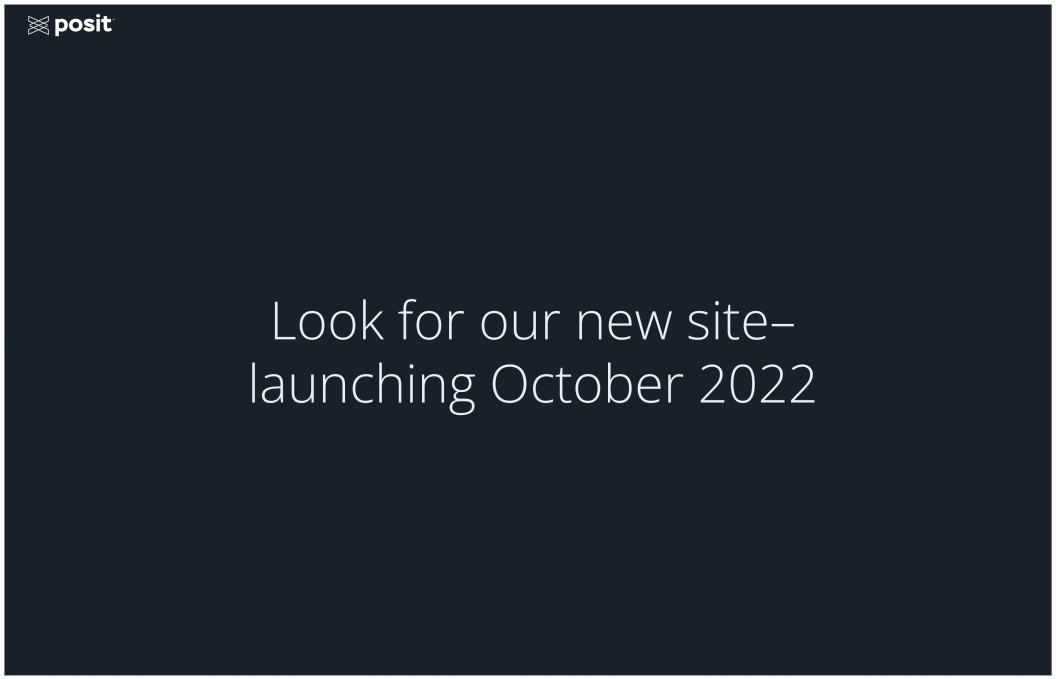
Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

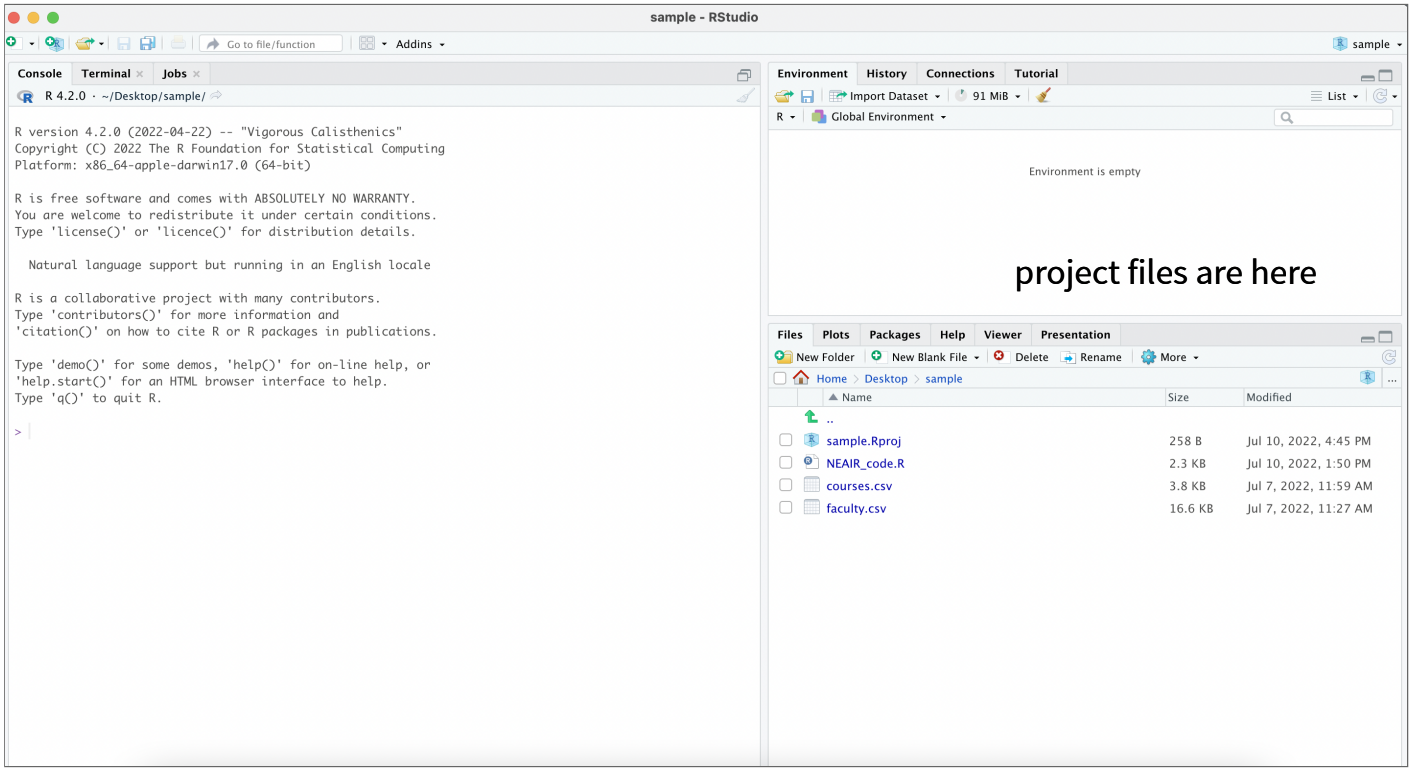
OS	Download	Size	SHA-256
Windows 10/11	RStudio-2022.07.2-576.exe	190.49 MB	b38bf925
macOS 10.15+	RStudio-2022.07.2-576.dmg	224.49 MB	35028d02
Ubuntu 18+/Debian 10+	rstudio-2022.07.2-576-amd64.deb	133.19 MB	b7d0c386
Ubuntu 22	rstudio-2022.07.2-576-amd64.deb	134.06 MB	e1c51003
Fedora 19/Red Hat 7	rstudio-2022.07.2-576-x86_64.rpm	103.29 MB	6594c7bf
Fedora 34/Red Hat 8	rstudio-2022.07.2-576-x86_64.rpm	150.13 MB	bcfce754
OpenSUSE 15	rstudio-2022.07.2-576-x86_64.rpm	134.10 MB	a266d996

RStudio = Posit in the near future

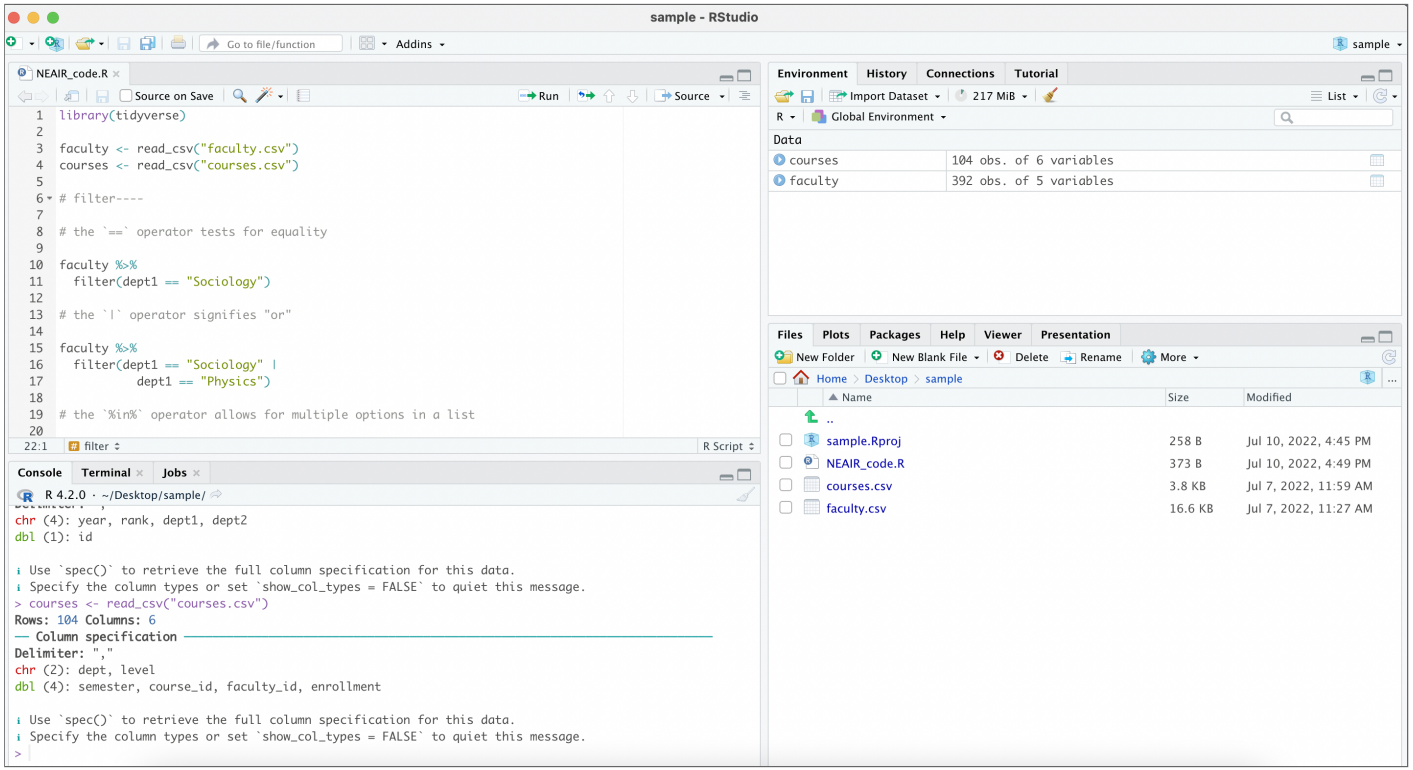
- open-source
- multi OS support
- in October 2022 new name: Posit



Navigating RStudio



Navigating RStudio



Using R

You can use R via *packages*

…which contain *functions*

…which are just *verbs*

Most valuable packages in natural heritage materials characterization

- tidyverse (ggplot2, dplyr, tidyr, purrr, readr, stringr)
- plotly
- shiny
- spectroscopy: hyperSpec, Chemospec
- recipes, H2O, lime, parsnip
- timetk, Modeltime
- knitr, RMarkdown, quarto

Dataset: Soil samples

`elements`

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr	Fe	K	Mg	Mn	Na
GER	8.8	604	l	18894	253	2437	25	5595	9804	905	217	2967
SKA	5.6	864	l	80340	862	5074	24	24690	21600	6272	620	15653
EST	5.8	579	l	40646	453	6833	20	12170	21625	3075	372	9273
LIT	6.1	619	l	34137	374	6032	25	10911	18927	3196	256	4377
NOR	-1.1	516	ll	43346	418	16231	22	15248	11082	4885	480	14169
PTG	16.9	670	s	101616	477	6997	83	44624	28673	13448	751	5045

Basic data manipulation

Useful operators

`<-`
“save as”
`Opt + -`
`Alt + -`
`%>%`
“and then”
`Cmd + Shift + m`
`Ctrl + Shift + m`

Common functions

`filter` keeps or discards rows (aka observations)
`select` keeps or discards columns (aka variables)
`arrange` sorts data set by certain variable(s)
`count` tallies data set by certain variable(s)
`mutate` creates new variables
`group_by`/`summarize` aggregates data (pivot tables!)
`str_*` functions work easily with text

Syntax of a function

```
function(data, argument(s))
```

is the same as

```
data %>%
  function(argument(s))
```

Filter

`filter` keeps or discards rows (aka observations)
the `==` operator tests for equality

```
1 elements %>%
2   filter(COUNTRY == "CRO")
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
CRO	10.8	1247	m	79017	440	4167	109
CRO	15.4	1315	m	69120	319	62501	334
CRO	11.1	674	s	74519	483	28938	96
CRO	8.2	1280	m	64939	372	5660	168
CRO	13.7	950	s	51867	291	14501	237
CRO	9.9	1301	m	72560	397	3445	124
CRO	13.0	1037	s	82087	420	11650	237
CRO	13.0	1140	s	59911	301	4310	228
CRO	10.9	898	s	78911	516	6904	91
CRO	11.2	680	s	66686	425	12093	94

Filter

the | operator signifies “or”

```
elements %>%
  filter(COUNTRY == "EST" |
           COUNTRY == "FIN")
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
EST	5.8	579	l	40646	453	6833	20
FIN	2.5	595	m	75524	741	15959	93
EST	5.2	617	l	39059	407	5911	17
FIN	5.2	658	m	71819	551	13622	75
FIN	1.3	622	ll	3758	85	13172	17
FIN	2.0	572	ll	62981	598	18082	21
FIN	-0.7	528	ll	52608	359	12450	92
FIN	2.8	615	l	61922	545	13444	65
FIN	-1.7	476	ll	1958	27	16510	8
FIN	5.4	606	ll	53084	425	9727	18

Filter

the %in% operator allows for multiple options in a list

```
elements %>%
  filter(COUNTRY %in% c("FRA",
                        "ITA",
                        "GER"))
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
GER	8.8	604	l	18894	253	2437	25
GER	9.1	528	s	43716	413	7340	54
ITA	17.2	517	s	65204	302	8562	90
FRA	11.3	759	m	84363	597	2551	77
ITA	12.4	887	m	66580	365	82905	56
GER	9.7	772	l	48903	242	4181	90
ITA	13.9	870	s	46945	341	139367	80
FRA	12.4	786	s	67427	336	4796	75
GER	7.4	637	m	76424	552	8362	66
FRA	11.6	755	m	54354	206	51944	79

Filter

the & operator combines conditions

```
elements %>%
  filter(COUNTRY %in% c("FRA",
                        "ITA",
                        "GER") &
           AnnPrec > 1000)
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
FRA	6.3	1042	m	58800	280	5460	68
GER	6.2	1194	m	86215	457	2080	118
ITA	12.1	1045	m	86268	520	22828	129
FRA	6.5	1152	m	65786	588	5410	62
FRA	10.3	1073	l	55836	280	3102	55
FRA	5.0	1182	l	38582	262	97414	40
ITA	12.2	1147	m	94418	683	21663	180
ITA	12.1	1328	m	45674	203	98057	42
ITA	12.3	1042	m	64939	418	7290	86
FRA	12.8	1190	m	89073	173	8634	169

Select

`select` keeps or discards columns (aka variables)

```
1 elements %>%
2   select(Country, MeanTemp, AnnPrec)
```

COUNTRY	MeanTemp	AnnPrec
GER	8.8	604
SKA	5.6	864
EST	5.8	579
LIT	6.1	619
NOR	-1.1	516
PTG	16.9	670
GER	9.1	528
LIT	5.9	627
POL	6.6	649
SWE	6.6	567

The Pipe

the pipe %>% chains multiple functions together

```
elements %>%
  select(COUNTRY, MeanTemp, AnnPrec) %>%
  filter(MeanTemp < 0)
```

COUNTRY	MeanTemp	AnnPrec
NOR	-1.1	516
FIN	-0.7	528
FIN	-1.7	476
SWE	-0.4	508
NOR	-1.2	543
NOR	-1.4	550
SWE	-0.5	542
FIN	-0.5	526
SWE	-1.9	462
SWE	-1.5	476

Arrange

`arrange` sorts data set by certain variable(s)
use `desc()` to get descending order

```
1 elements %>%
2   arrange(desc(AnnPrec))
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
NOR	6.8	2561	ll	63722	822	14666	16
NOR	5.9	2560	ll	69597	1149	17839	12
NOR	5.0	2544	ll	72084	421	36228	146
NOR	7.4	2364	ll	45039	512	13179	69
NOR	5.2	2344	l	64569	301	22913	71
NOR	7.3	2291	ll	13390	101	6518	29
NOR	4.6	2259	l	70602	313	7569	113
NOR	6.1	2241	ll	78964	1418	13251	11
NOR	7.2	2122	ll	57212	207	20105	90
NOR	6.3	2009	ll	65257	674	25672	66

Arrange

can sort by multiple variables

```
elements %>%
  arrange(Country, desc(AnnPrec))
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	Ca	Cr
AUS	5.0	1366	l	71237	395	14094	93
AUS	7.0	1277	m	61499	380	50136	81
AUS	8.7	1264	l	90714	695	6139	100
AUS	7.9	1257	m	50067	298	4031	94
AUS	8.9	1201	m	68908	355	4224	118
AUS	8.5	1136	m	43346	279	4803	75
AUS	9.0	1126	m	46415	196	90481	68
AUS	5.7	1124	m	89020	705	17439	122
AUS	8.8	1119		40964	160	85979	66
AUS	7.4	1115	m	80870	616	4881	90

Count

`count` tallies data set by certain variable(s) (very useful for familiarizing yourself with data)

```
1 elements %>%
2   count(COUNTRY)
```

COUNTRY	n
AUS	35
BEL	13
BOS	16
BUL	44
CRO	29
CYP	6
CZR	33
DEN	17
EST	17
FIN	148

Count

can use `sort = TRUE` to order results

```
1 elements %>%
2   count(COUNTRY, soilclass, sort = TRUE)
```

COUNTRY	soilclass	n
FRA	m	89
SPA	m	87
UKR	m	73
FIN	ll	72
FRA	l	70
NOR	ll	70
GER	m	69
SWE	l	65
SWE	ll	64
UNK	l	60

Mutate

`mutate` creates new variables (with a single `=`)

```
elements %>%
  mutate(logZn = log10(Zn)) %>%
  select(COUNTRY, Zn, logZn)
```

COUNTRY	Zn	logZn
GER	30	1.477121
SKA	67	1.826075
EST	59	1.770852
LIT	33	1.518514
NOR	41	1.612784
PTG	129	2.110590
GER	74	1.869232
LIT	36	1.556302
POL	39	1.591065
SWE	22	1.342423

Group by / summarize

`group_by`/`summarize` aggregates data (pivot tables!)
`group_by()` identifies the grouping variable(s) and `summarize()` specifies the aggregation

```
elements %>%
  group_by(COUNTRY, soilclass) %>%
  summarize(MeanCountryTemp = mean(MeanTemp))
```

COUNTRY	soilclass	MeanCountryTemp
AUS		8.800000
AUS	l	7.930000
AUS	m	8.336364
AUS	s	9.200000
BEL	l	8.975000
BEL	ll	10.000000
BEL	m	9.614286
BOS	l	9.000000
BOS	m	10.190000
BOS	s	11.525000

Group by / summarize

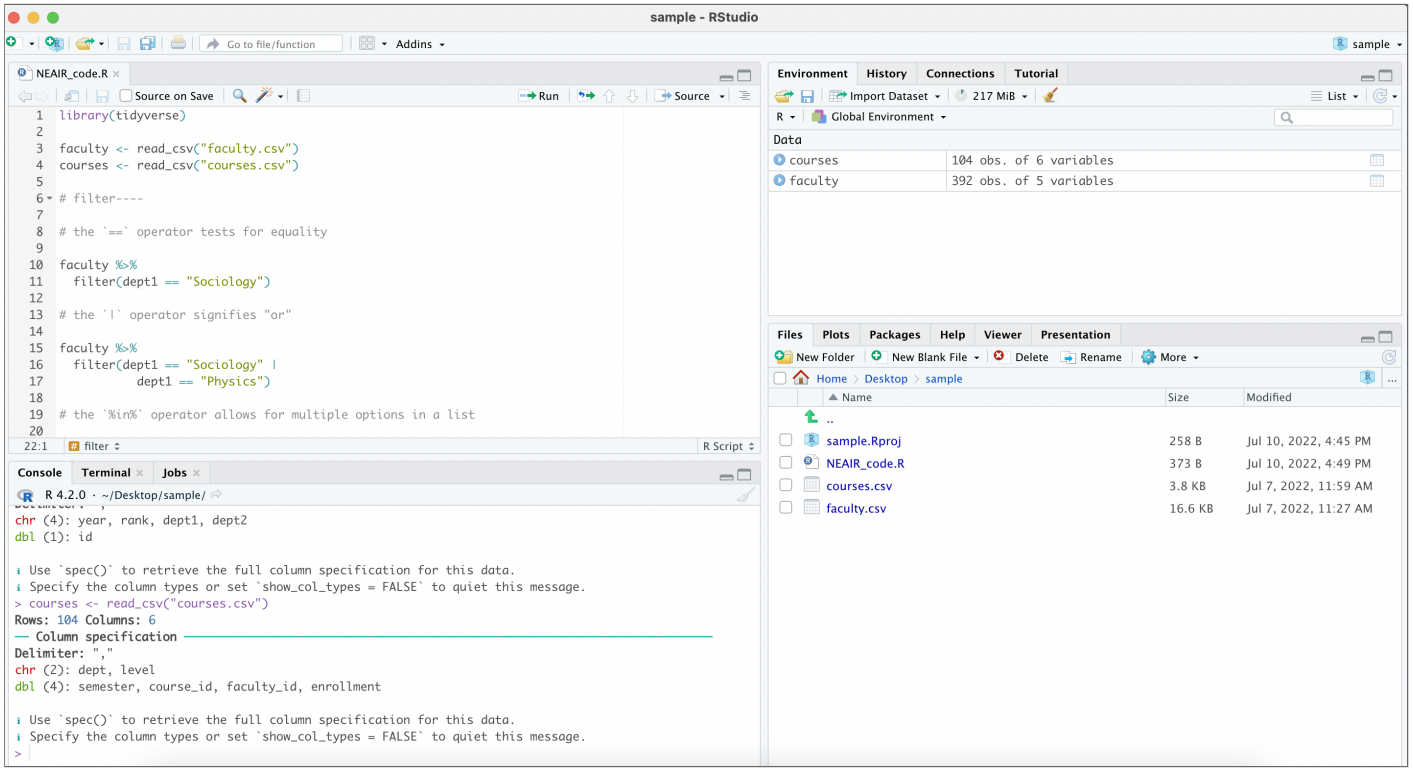
useful arguments within `summarize`:
`mean`, `median`, `sd`, `min`, `max`, `n`

```
elements %>%
  group_by(COUNTRY) %>%
  summarize(minAP= min(AnnPrec),
            maxAP = max(AnnPrec))
```

COUNTRY	minAP	maxAP
AUS	622	1366
BEL	670	1048
BOS	798	1327
BUL	470	672
CRO	674	1377
CYP	349	975
CZR	527	914
DEN	572	871
EST	579	708
FIN	391	667

Using R effectively

Working in RStudio



Working in RStudio

Typing in the console

- useful for quick notes but disposable
- actions are saved but code is not (except history mechanism)
- one chunk of code is run at a time (`Return`, `Enter`)

Typing in a code file

- script files have a `.R` extension
- code is saved and sections of any size can be run (`Cmd + Return`, `Ctrl + Enter`)
- do ~95% of your typing in a code file instead of the console!

imported data shows up here

Working with packages

packages need to be installed once on each computer you use

```
# only need to do this once (per computer)
install.packages("tidyverse")
```

packages need to be loaded/attached with `library()` at the beginning of every session

```
# always put the necessary packages at the top of a code file
library(tidyverse)
library(hyperSpec)
```

access help files by typing `??tidyverse` or `??mutate` in the console

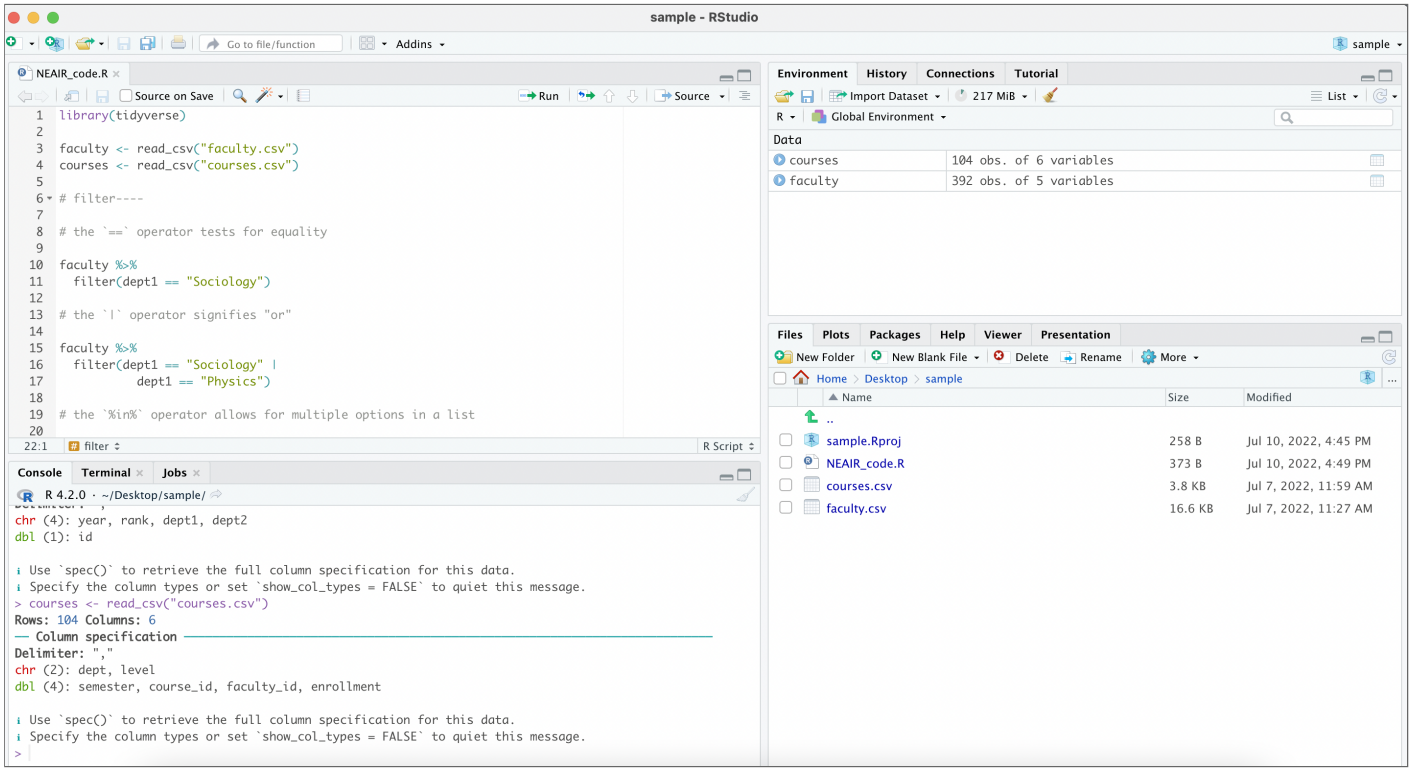
Organizing with projects

highly recommend using **projects** to stay organized

keeps code files and data files together, allowing for easier file path navigation and better reproducible work habits

```
File -> New Project
```

Organizing with projects



project files are here

Accessing data

use `read_csv()` to import a csv file

```
# the file path is this simple if you use projects!
# ?read_csv() in the console will bring up the help file with more options
faculty <- read_csv("elements.csv")
```

the **readxl** package is helpful for Excel files

```
# needs to be loaded but not installed as it's part of the tidyverse
library(readxl)
faculty <- read_excel("elements.xlsx", sheet = 1)
```

view the data with `View(elements)` or by clicking on the data name in the Environment pane

Stringr functions

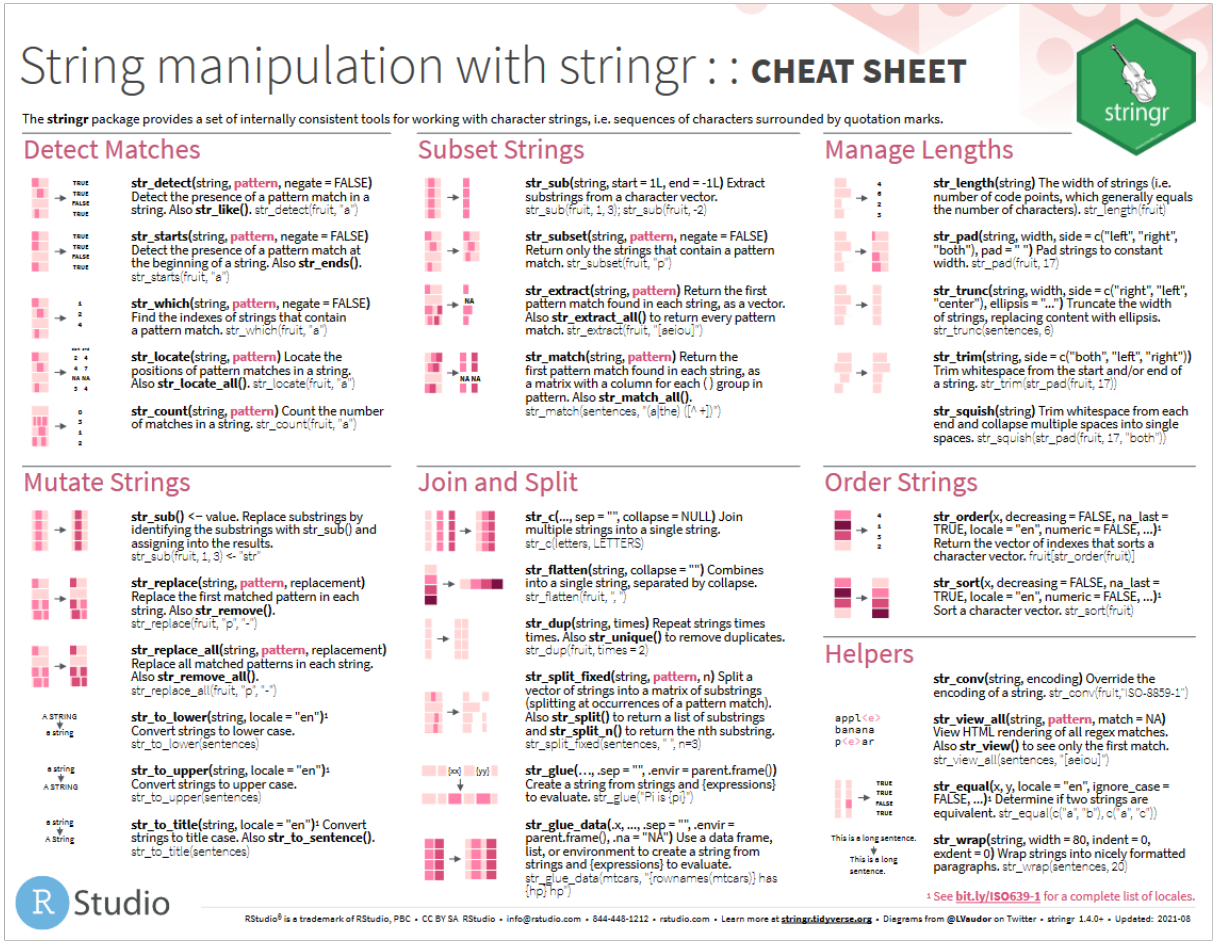
functions from `stringr` (which all start with `str_`) are useful for working with text data

```
elements%>%
  filter(str_detect(soilclass, "l"))
```

COUNTRY	MeanTemp	AnnPrec	soilclass	Al	Ba	
GER	8.8	604	l	18894	253	24
SKA	5.6	864	l	80340	862	50
EST	5.8	579	l	40646	453	68
LIT	6.1	619	l	34137	374	60
NOR	-1.1	516	ll	43346	418	162
SWE	6.6	567	l	53560	424	74
DEN	7.7	833	ll	7992	112	11
UKR	7.1	592	ll	10320	150	8
NOR	5.7	1763	l	72349	369	213
GER	9.7	772	l	48903	242	41

Stringr functions

https://www.rstudio.com/resources/cheatsheets/ R Studio Cheatsheets



String manipulation with stringr : : **CHEAT SHEET**

The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

Detect Matches

str_detect(string, **pattern**, negate = FALSE) Detect the presence of a pattern match in a string. Also **str_like()**. str_detect(fruit, "a")

str_starts(string, **pattern**, negate = FALSE) Detect the presence of a pattern match at the beginning of a string. Also **str_ends()**. str_starts(fruit, "a")

str_which(string, **pattern**, negate = FALSE) Find the indexes of strings that contain a pattern match. str_which(fruit, "a")

str_locate(string, **pattern**) Locate the positions of pattern matches in a string. Also **str_locate_all()**. str_locate(fruit, "a")

str_count(string, **pattern**) Count the number of matches in a string. str_count(fruit, "a")

Subset Strings

str_sub(string, start = 1L, end = -1L) Extract substrings from a character vector. str_sub(fruit, 1, 3); str_sub(fruit, -2)

str_subset(string, **pattern**, negate = FALSE) Return only the strings that contain a pattern match. str_subset(fruit, "p")

str_extract(string, **pattern**) Return the first pattern match found in each string, as a vector. Also **str_extract_all()** to return every pattern match. str_extract(fruit, "[aeiou]")

str_match(string, **pattern**) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all()**. str_match(sentences, "(a|the) ([^ +])")

Manage Lengths

str_length(string) The width of strings (i.e. number of code points, which generally equals the number of characters). str_length(fruit)

str_pad(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. str_pad(fruit, 17)

str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis. str_trunc(sentences, 6)

str_trim(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. str_trim(str_pad(fruit, 17))

str_squish(string) Trim whitespace from each end and collapse multiple spaces into single spaces. str_squish(str_pad(fruit, 17, "both"))

Mutate Strings

str_sub() <– value. Replace substrings by identifying the substrings with str_sub() and assigning into the results. str_sub(fruit, 1, 3) <- "str"

str_replace(string, **pattern**, replacement) Replace the first matched pattern in each string. Also **str_remove()**. str_replace(fruit, "p", "-")

str_replace_all(string, **pattern**, replacement) Replace all matched patterns in each string. Also **str_remove_all()**. str_replace_all(fruit, "p", "-")

str_to_lower(string, locale = "en")[1] Convert strings to lower case. str_to_lower(sentences)

str_to_upper(string, locale = "en")[1] Convert strings to upper case. str_to_upper(sentences)

str_to_title(string, locale = "en")[1] Convert strings to title case. Also **str_to_sentence()**. str_to_title(sentences)

Join and Split

str_c(..., sep = "", collapse = NULL) Join multiple strings into a single string. str_c(letters, LETTERS)

str_flatten(string, collapse = "") Combines into a single string, separated by collapse. str_flatten(fruit, ", ")

str_dup(string, times) Repeat strings times times. Also **str_unique()** to remove duplicates. str_dup(fruit, times = 2)

str_split_fixed(string, **pattern**, n) Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split()** to return a list of substrings and **str_split_n()** to return the nth substring. str_split_fixed(sentences, " ", n=3)

str_glue(..., .sep = "", .envir = parent.frame()) Create a string from strings and {expressions} to evaluate. str_glue("Pi is {pi}")

str_glue_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")

Order Strings

str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Return the vector of indexes that sorts a character vector. fruit[str_order(fruit)]

str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Sort a character vector. str_sort(fruit)

Helpers

str_conv(string, encoding) Override the encoding of a string. str_conv(fruit,"ISO-8859-1")

str_view_all(string, **pattern**, match = NA) View HTML rendering of all regex matches. Also **str_view()** to see only the first match. str_view_all(sentences, "[aeiou]")

str_equal(x, y, locale = "en", ignore_case = FALSE, ...)[1] Determine if two strings are equivalent. str_equal(c("a", "b"), c("a", "c"))

str_wrap(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. str_wrap(sentences, 20)

[1] See bit.ly/ISO639-1 for a complete list of locales.

RStudio® is a trademark of RStudio, PBC • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at stringr.tidyverse.org • Diagrams from @LVaudor on Twitter • stringr 1.4.0+ • Updated: 2021-08

Data visualization

ggplot2

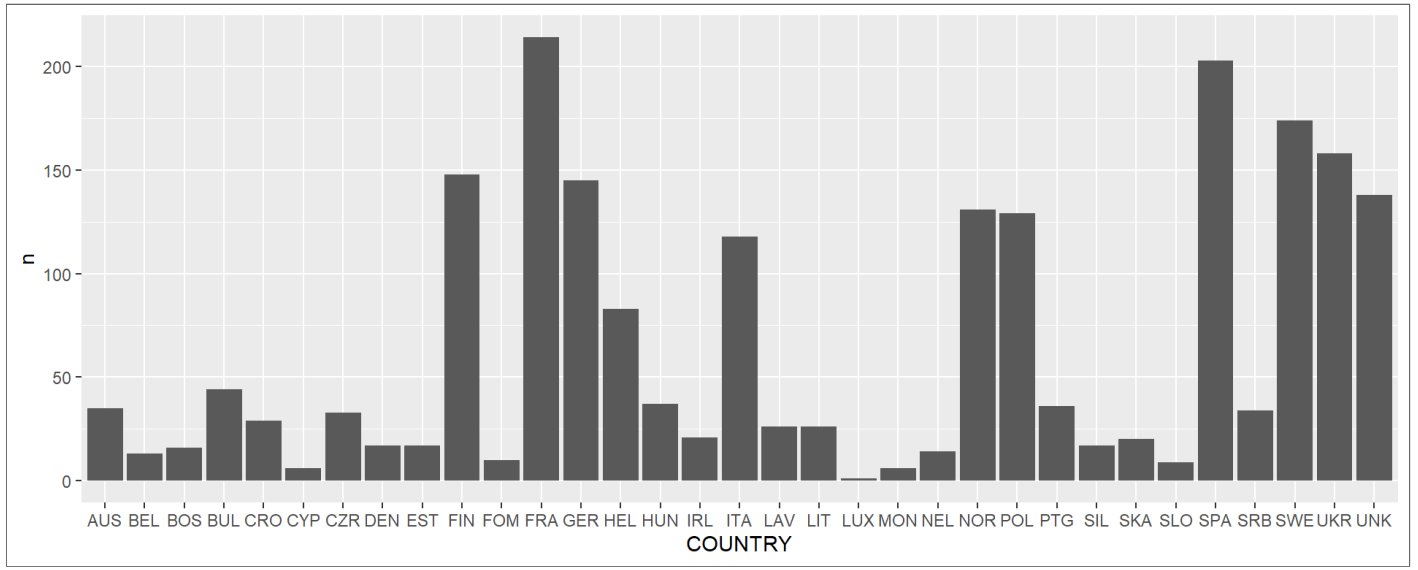
`ggplot2` is the data visualization package that is loaded with the `tidyverse`

the grammar of graphics maps data to the aesthetic attributes of geometric points

encoding data into visual cues (e.g., length, color, position, size) is how we signify changes and comparisons

Bar chart

```
elements %>%
  count(COUNTRY) %>%
  ggplot(aes(x = COUNTRY, y = n)) +
  geom_bar(stat = "identity")
```

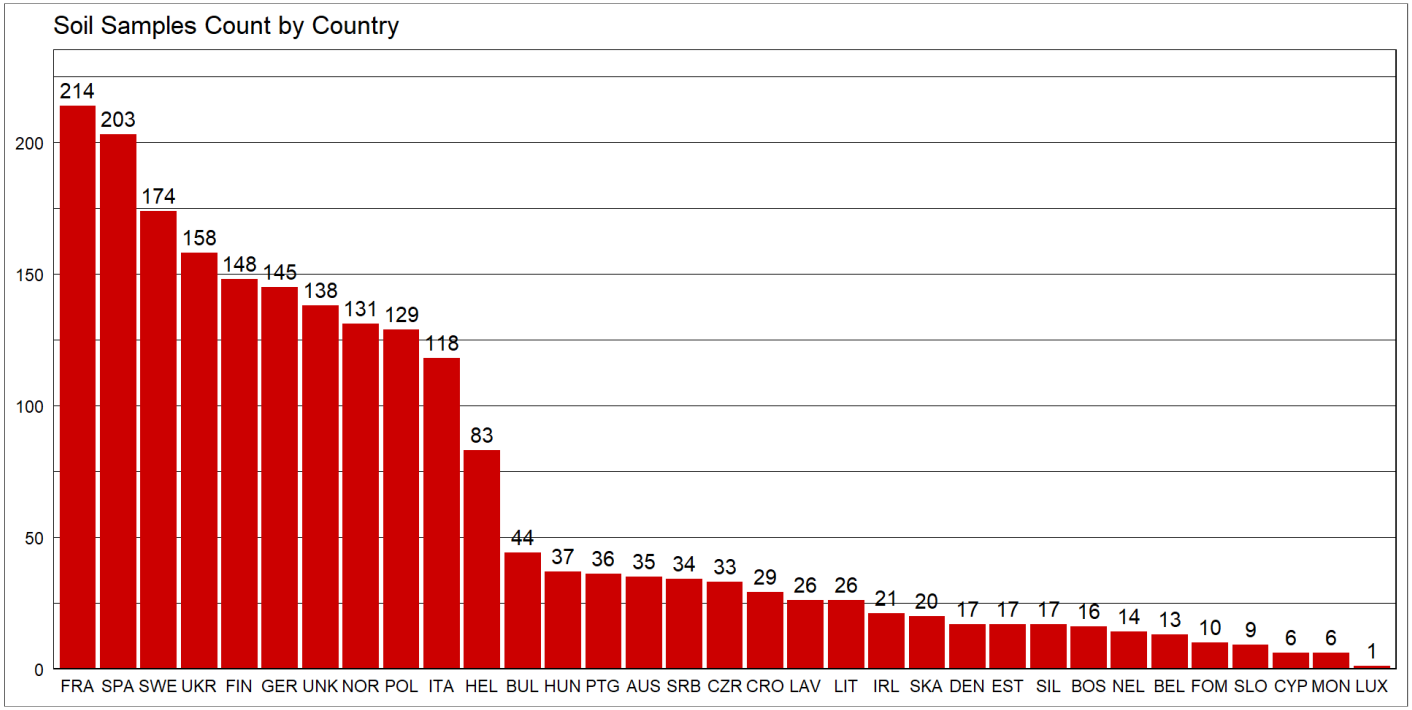


to combine lines into one code chunk, use + instead of %>%

Bar chart

We can create a prettier plot pretty easily

▶ expand for full code



Putting it all together

With what we've done so far, your `.R` file could:

- import your data files
- document all data cleaning and preparation steps and decisions
- produce presentation ready graphic summarizing your results

and that file would make it extremely easy for you or someone else to reproduce this analysis with new data in the future

R Markdown

using RStudio, create `.Rmd` documents that combine text, code, and graphics

many output formats: html, pdf, Word, slides

exceedingly useful for parameterized reporting: can create an R-based PDF report and generate it automatically for, say, each department

R Markdown


Rmarkdown
TEXT. CODE. OUTPUT.
(GET IT TOGETHER, PEOPLE.)
#HEADERS
CODE
TEXT
FIGURE
PANDOC
KNITR
FAIRY DUST
RECIPE
1. ADD TEXT
2. ADD CODE
3. KNIT
4. (magic)
5. CELEBRATE PERCEIVED WIZARDRY
YAML
rmarkdown
.Rmd
knitr
pandoc

Internal packages

you can also create your *own* packages!

your package can hold:

- common data sets that are used across projects
- custom `ggplot2` themes
- common functions and calculations (and their definitions!)
- can be stored on a shared drive to facilitate collaboration

Focus on: OPEN SCIENCE

Open Science

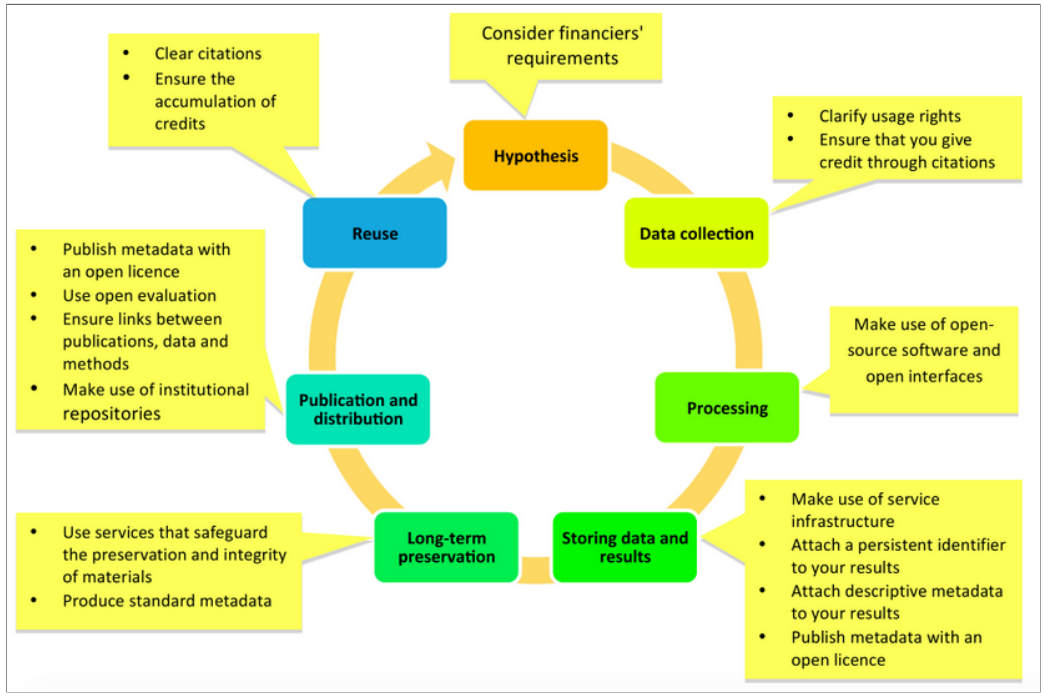
Important

open science is defined as an inclusive construct that combines various movements and practices aiming to make multilingual scientific knowledge openly available, accessible and reusable for everyone, to increase scientific collaborations and sharing of information for the benefits of science and society… It builds on the following key pillars: open scientific knowledge, open science infrastructures, science communication, open engagement of societal actors and open dialogue with other knowledge systems.

Open Science

- transparency and repeatability
- data formats
- data repositories

Workflow



Promoting openness at different stages of the research process (Open Science and Research Initiative, 2014)

Learn more about R

General

https://r4ds.had.co.nz R for Data Science the ultimate guide
https://rstudio-conf-2020.github.io/r-for-excel R for Excel users
https://stat545.com) an online book on reproducible data analysis in R
https://education.rstudio.com RStudio Education

R Markdown and packages

R Markdown

https://rmarkdown.rstudio.com/ The official R Markdown website

https://bookdown.org/yihui/rmarkdown R Markdown: The Definitive Guide

internal packages

https://emilyriederer.netlify.app/post/team-of-packages A comprehensive theoretical explainer

https://www.tidyverse.org/learn Learn `tidyverse`

ggplot2 resources

https://r4ds.had.co.nz/data-visualisation.html R for Data Science
https://socviz.co/index.html Data Visualization: a practical introduction
https://ggplot2-book.org/ ggplot2 book
https://r-graph-gallery.com/ R graph gallery

Take Home Message

- R Environment open source multi-OS solution for data science
- `tidyverse` packages for data wrangling
- `hyperSpec` and `Chemospec` for spectral data processing
- `ggplot2` for visualisation
- `RMarkdown`, `Quarto` for reports, presentations, articles, books, websites
- Repeatability and transparency of code-driven analysis -open science
- Huge library of free materials to learn and master R related skills

My last secret

The goal of the `here` package is to enable easy file referencing in project-oriented workflows. In contrast to using `setwd()`, which is fragile and dependent on the way you organize your files, `here` uses the top-level directory of a project to easily build paths to files.

THANK YOU FOR YOUR ATTENTION