**20.10.22**

Console

* Type random letter -> error, because it interprets letter as objects / variables -> a sentence about errors
* Type number -> R gets number -> add numbers together -> spaces do not matter -> combine spaces with letters -> error
* Arrow up

Editor

* Load and save script -> why? -> save what you have programmed -> execute scripts for new data
* Assign variables: using „<-“ not „=“ -> appears in environment -> muster start with letter, can only use letter,number,\_,. -> no spaces -> capitalization matters -> tab to autocomplete (faster & avoids typos)
* Assign variables: use decriptive names -> not x, y
* Execute command in editor -> not just enter -> Run/Source/Cmd/Ctrl-Enter -> Source executes whole scripts one line at a time -> variables must be declared before (clear environment to demonstrate)
* Comment code -> like writing a labbook -> explain what the code below is doing (and maybe why – or say why in the markdown)

Markdown

* Markdown is a lightweight markup language
* It is designed to be easy to write using any generic text editor -> generates nicles formatted text
* YAML Header -> specifies global Document options
* Code belongs in code chunks

**27.10.22**

Load in data

* For that, we need a function -> what is a function:
* R has a large collection of built-in functions (little programs/code pieces) that are called like this:
* function\_name(arg1 = val1, arg2 = val2, ...) -> explain argument
* a lot of what we will do here is applying different functions
* read.csv() -> how do we use it? -> help(read.csv) / ?read.csv
* working dir, setwd(), getwd()
* read in data using a long variable\_name -> show tab for autocomplete
* View(), head(), tail()
* str() -> explain datatypes in R: num, char, factor etc -> explain as.character, as.integer …
* class() for finding out what datatype your variable is

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

* if time -> load dyplr package -> explain what a library is (collection of functions) -> dplyr awesome for table manipulation -> demonstrate filter() function
* explain concept of pipe %>%

**02.11.22**

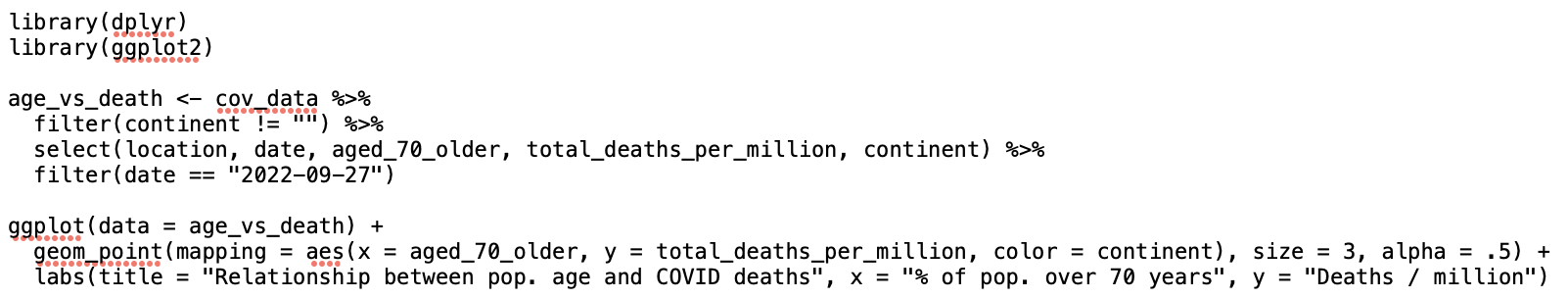
Dplyr

* recap -> introduce the View() function
* what is a library? Install dplyr -> load dplyr
* filter() -> talk about library conflicts (multiple libraries can have a function called “filter”)
  + Example: location == “Germany” -> show typo (e.g. location == “germany”)
  + Example: filter(total\_cases == 4, continent == “Europe”) -> show that the order matters
  + Save new DF as a new variable
  + Logical operators: |, >, >= etc
* select() function
* count() function
* rename() function -> rename the ‘n’ columns after applying count()
* introduce pipes -> what if you want to combine multiple functions -> introduce simplified logic at the board by hand drawing -> then show code

**09.11.22**

Ggplot2

* introduce ggplot2 package -> explain versatility
* install ggplot2 -> load -> remind / explain that you only need to install once, but load every time
* ggplot() function creates a coordinate system that you can add layers to -> first argument is the data
* An aesthetic is a visual property of the data points in your plot -> powerful feature is that variables can be mapped to visual properties
* geom functions that each add a different type of layer to a plot
* talk about the %>% + problem
* go through example:
  + 1st: explain the locations == “Continent” problem and filter(continent != “”) -> explain that I always make a new line (although not necessary -> but R does not mind and I think it makes it more readable)
  + 2nd: select location, date, aged\_70\_older, total\_deaths\_per\_million, continent (to make it look more tidy -> not really necessary)
  + 3rd: look at only one date
  + Perform gglot layer by layer -> add color, size, shape, alpha
  + Let them explore outliers (using filter() function)
  + If there is time: introduce labs() function



**24.11.22**

Cycle2

* Introduce data type “date”
* Mutate(date = as.Date(date)) -> explain that mutate with the same column name will overwrite
* Download and read in the data
* Get only the newest data: filter(Year\_FUBC\_publication == 2022)
* summarise() -> collapses a data frame to a single row -> show example summarise(median(Crop\_area\_k\_ha)) -> introduce mean(), median() function -> can be named: median\_crop\_size = median(Crop\_area\_k\_ha)
* not terribly useful unless we pair it with group\_by()
* group\_by(Crop) -> now median\_crop\_size / Crop
* group by two variables: group\_by(Crop, Region\_IFA)
* summarise: include n() -> counts all members of the group

**01.12.22**

Stringr und regex

* homework brief recap -> ask if it is ok if I put student’s markdown on cloud?
* What is the most common crop?
  + fert\_data %>% count(Crop) %>% arrange(desc(n)) %>% head(3)
* Problem: There are many entries containing Maize that are not exactly called "Maize"
  + fert\_data %>% count(Crop) %>% View()
* How can we filter for all rows containing the word maize (not exactly “maize) -> intro lib(stringr)
  + Library(stringr)
  + fert\_data %>% filter(str\_detect(string = Crop, pattern = "Maize")) %>% count(Crop)
* Problem: "maize" is not inside because no capital "M" -> intro regex
* Matching patterns with regular expressions: regex = language to describe patterns in strings
* For learning regex: let’s step away from our dataset for something more simple -> will help illustrate
* What is in a column is a vector -> create simple vector using c() function -> c(“apple”, banana”, “pear”)
* str\_detect() -> on “fruit” test\_string -> explain TRUE / FALSE and why/how it works together with filter
  + basic matches: match “an” (capitalization matters) -> “.” Matches any character

str\_detect(string = test\_string, pattern = "a")

str\_detect(string = test\_string, pattern = "ap")

"." -> any character

str\_detect(string = test\_string, pattern = "ap.le")

* + anchors: str\_detect(x, “^a) -> “a” at the beginning, str\_detect(x, “a$”) -> “a” in the end

str\_detect(string = test\_string, pattern = "^a")

str\_detect(string = test\_string, pattern = "a$")

* + multiple matches: There are a number of special patterns that match more than one character. You’ve already seen “.”
  + [abc]: matches a, b, **or** c.

str\_detect(string = test\_string, pattern = "[ap]p")

x <- c("apple", "Apple")

str\_detect(string = x, pattern = "[Aa]pple")

str\_detect(string = x, pattern = ".pple")

* Let’s fix our [Mm]aize problem

firt\_data %>% filter(str\_detect(string = Crop, pattern = "[Mm]aize")) %>% count(Crop)

* Analyse Germany's maize production over time?

firt\_data %>%

filter(str\_detect(string = Crop, pattern = "[M|m]aize")) %>%

filter(Country == "Germany") %>%

group\_by(Year) %>%

summarise(maize\_area = sum(Crop\_area\_k\_ha)) %>%

ggplot(aes(x = Year, y = maize\_area, group = 1)) +

geom\_point() +

geom\_line()

**08.12.22**

pivot\_longer and facets

* Homework recap -> introduce repetitions
* Rewind to homework from week6 -> fertilizer consumption of **china and india** over time -> show plot you have in mind on white board
* fert\_data %>% filter(Country == "India" | Country == "Germany") %>% group\_by(Year, Country) %>% summarise(n = sum(N\_k\_t), p = sum(P2O5\_k\_t), k = sum(K2O\_k\_t))
* problem: data in a funny (wide) format -> what would we use for the y-axis??
* I was hiding something from students: all data we worked with was in so-called “long format” aka tidy format
  + What is the long format? Each column is a variable and each row is an observation/value
  + sometimes data is not: example rows = countries, cols = different years (1998, 1999 …), cells = cases or something
  + column names are not names of variables, but values of a variable!
* In long format that would translate to rows = countries, cols = “year”, “cases”, cells = 1999, case number etc
* pivot\_longer(cols = c(n,p,k), names\_to = "fertilizer\_type", values\_to = "kilotonnes")
* facets: subplots that each display one subset of the data -> facet\_wrap()
* code for plot: ggplot(mapping = aes(x = Year, y = kilotonnes, group = fertilizer\_type)) + geom\_point(aes(color = fertilizer\_type)) + geom\_line() + facet\_wrap(~ Country)