GRAPH Network

Further data analysis with R

Strengthen your public health data analysis skills

Book developed by the GRAPH Network, with support from the World Health Organization

Further data analysis with R (draft)

A guidebook for analyzing public health data with R

June 2022

This document serves as a course accompaniment for the 'Further data analysis with R' course that can be found on https://thegraphcourses.org.
The GRAPH Courses is a project of the Global Research and Analyses for Public Health (GRAPH) Network, with the support of the World Health Organization (WHO).

Table of contents

1

Date/time data: introduction

1.1 Learning Objectives

1. Learn about Build a date object by providing individual components (year, month, day) using `{clock}` and the `date_build()` function.
2. Create a date-time object by providing individual components (year, month, day, hour, minute, second) using `{clock}` and the `date_time_build()` function.

1.2 Overview

As a data analyst, being able to work with temporal data (that is data associated with dates and times) is a key skill to have.

Temporal data can be inputted in several ways. For example, when recording the date a patient visited the clinic, one nurse could input the date this way - Monday, 3rd January, 2022 while another nurse could record the same date as 03/01/2022. Thus, if our aim is to find out how many patients visited the hospital on a particular day, the data must be cleaned and the columns containing the date and/or time information must be reformatted.

1.2.1 Date and time classes in R

Here are some fundamental things to know about working dates and times in R.

R has special classes to deal with date and/or time data. We are going to discuss three of them.

- The `date` class for dates (which can be formatted into different configurations of year-month-day.
- The `time` class for times (hours, minutes and seconds)
- The `date-time` class also called POSIXct used to represent date plus time that is to pin point a distinct moment in time on a particular day in specified time zone.

For example, to get the current date and time in your local time zone, R has two in-built functions.

`Sys.Date()` for the current date and `Sys.time()` for the current date-time in the time zone on your computer.

```
Sys.time()
```

```
## [1] "2022-06-22 12:35:44 CEST"
```

```
Sys.Date()
```

```
## [1] "2022-06-22"
```

1.3 Creating date and times

Clock, is an R package that has many functions that make working with dates and times easier.

To install `{clock}`, run the following code in your R session.
`install.packages("clock")`.

Proceed to load the packages if you have already installed them.

1.3.1 The `date_build()` function

We are going to use the function `date_build()` in the `{clock}` package to date objects with the birthdays given in the report above.

`clock::date_build()` is a function used to create a date by providing the following arguments.

- The first argument provided is the year. If only one argument is given, the date output is the first day of the first month of that year.

  For example, run the code below, to quickly get the date- the first of January, 2022.

```
clock::date_build(2022)
```

```
## [1] "2022-01-01"
```

You can specify the year, month and day by providing three integer values separated by commas. For example, the output of the following code is 15th March, 2022.

```
clock::date_build(2022,3,15)
```

```
## [1] "2022-03-15"
```

- The second argument of the `clock::date_build()` function has a range of 1 to 12 to represent the 12 months (January to December).
- The third argument of this function has a range of 1 to 31 to represent the number of possible days in a month.

Thus, providing values outside these ranges would be invalid. For example, the code below gives an error because there is no 14th month or 40th day of the year 2007.

```
try(clock::date_build(2007,14))
```

```
## Error : `month` must be within the range of [1, 12], not 14.
```

```
try(clock::date_build(2007,12,40))
```

```
## Error : `day` must be within the range of [1, 31], not 40.
```

You can create a range of dates using the colon `:` operator between the range you want either in the year, month or day arguments.

For example, to represent the first week of February 2002, in place of the third argument (day) in `clock::date_build(year, month, day)`,type `1:7` as shown below.

```
date_build(2002, 2, 1:7)
```

```
## [1] "2002-02-01" "2002-02-02" "2002-02-03" "2002-02-04" "2002-02-05"
"2002-02-06" "2002-02-07"
```

Also, to get the last day of each month the year 2015, you can type the following code in place of `clock::date_build(year,month, day)` .

- The month argument is replaced by the range `1:12`
- Then, the day argument is replaced by the string `"last"`

```
clock::date_build(2015,1:12,"last")
```

```
##  [1] "2015-01-31" "2015-02-28" "2015-03-31" "2015-04-30" "2015-05-31"
"2015-06-30" "2015-07-31" "2015-08-31" "2015-09-30" "2015-10-31"
## [11] "2015-11-30" "2015-12-31"
```

Supposing you are analyzing birth records from a hospital in Nairobi, Kenya in the report below.

> A healthy infant male was born on 27th December 2021 at 6 am. The next day, 3 baby girls were delivered. They were born in 1 hour intervals from 3pm.

1. Create date objects from this information and write code to output the baby boy's correct birthday.

1.3.2 The `date_time_build()` function

To create a date-time object, a R object that contains both dates and times, we use the function `date_time_build()` with seven arguments `(year, month, day, hour, minute, second, zone)`.

A key thing, to note is how to input the arguments of date_time_build(). As discussed previously, month and day have specified ranges, so do hour, minute, second and zone.

- The hour argument ranges from 0-23
- The minute argument ranges from 0-59
- The second argument ranges from 0-59
- The zone is the local timezone written in the form "continent/city" eg. "America/New York" or "Africa/Lagos". You can check out the list of all time zones with associated cities by viewing this inbuilt R data set `OlsonNames()`

```
head(OlsonNames()) #Get first 6 rows of the timezones data set
```

```
## [1] "Africa/Abidjan"     "Africa/Accra"       "Africa/Addis_Ababa"
"Africa/Algiers"     "Africa/Asmara"      "Africa/Asmera"
```

For example, to create a date-time object of the boy born on 27th December, 2021 at 6 am in Nairobi, Kenya we should run the following code.

```
clock::date_time_build(2021,12,27,06,00,00, zone="Africa/Nairobi")
```

```
## [1] "2021-12-27 06:00:00 EAT"
```

2. Create a data object for the birthdays of the three baby girls born the following day after 27th December, 2021 at 3pm with one hour interval between them.

```
##
##   1   2
## Yowza! OH!-OH! This is frankly striking!
```

Contributors

The following team members contributed to this lesson:

AMA KORANTEMA OWUSU-DARKO

Data Analyst
Passionate about equipping health professionals with data analysis skills to promote evidence based research.

References

Some material in this lesson was adapted from the following sources:

- Wickham, Hadley, and Garrett Grolemund. "R For Data Science." 16 Dates and times | R for Data Science, 2017. https://r4ds.had.co.nz/dates-and-times.html#time-zones.
- Vaughan, Davis. "Building: Date-Time - date_time_build." - date_time_build • clock, February 12, 2021. https://clock.r-lib.org/reference/date_time_build.html.

2

Date/time data: disaggregating dates

2.1 Learning Objectives

1. Group temporal data into different categories (days, weeks, months and years)

When working with data is important for data analysts to understand that data aggregation can hide a lot of things.

For example, when you rely on monthly case counts to tell you the entire story about the development of new disease cases, there is a lot granular data you are not taking advantage of.

Disaggregating (separating) monthly data into weekly case counts and even daily counts might reveal more information about the progression of diseases, or trend of sales of a product.

We are going to use the ebola_sierraleone_2014 data set from the `{outbreaks}` package to explore historical Ebola outbreak data and demonstrate this concept.

Run the code chunk below to create an R object called ebola_data. With `head(ebola_data)` function, you can explore the initial rows of the data set.

```
#Create an R object named ebola_data
ebola_data <- outbreaks::ebola_sierraleone_2014
head(ebola_data) #show the first 6 rows of the ebola_data
```

```
##   id age sex    status date_of_onset date_of_sample district    chiefdom
## 1  1  20   F confirmed    2014-05-18     2014-05-23 Kailahun  Kissi Teng
## 2  2  42   F confirmed    2014-05-20     2014-05-25 Kailahun  Kissi Teng
## 3  3  45   F confirmed    2014-05-20     2014-05-25 Kailahun Kissi Tonge
## 4  4  15   F confirmed    2014-05-21     2014-05-26 Kailahun  Kissi Teng
## 5  5  19   F confirmed    2014-05-21     2014-05-26 Kailahun  Kissi Teng
## 6  6  55   F confirmed    2014-05-21     2014-05-26 Kailahun  Kissi Teng
```

The `names(ebola_data)` function would show the names of the different columns.

```
names(ebola_data)
```

```
## [1] "id"              "age"             "sex"             "status"
"date_of_onset"   "date_of_sample" "district"        "chiefdom"
```

It seems intuitive that `ebola_data$date_of_onset` and `ebola_data$date_of_sample` are columns containing date information. Let us confirm that by checking the class of the columns.

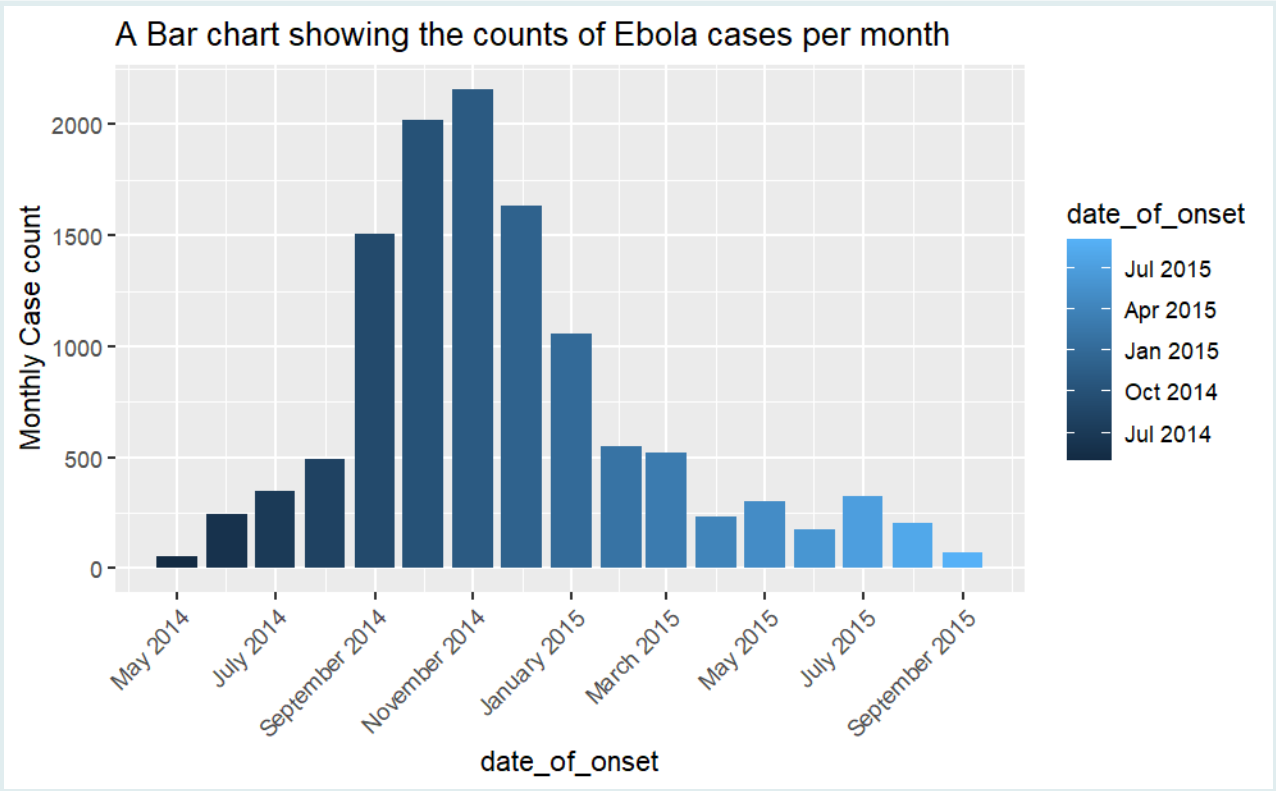
```
class(ebola_data$date_of_onset)
```

```
## [1] "Date"
```

```
class(ebola_data$date_of_sample)
```

```
## [1] "Date"
```

Here is a monthly breakdown of a past epidemic of Ebola. We see that there is a peak in October-December.



To create the bar chart of monthly totals of new Ebola cases above, we would use the `floor_date()` from the `{lubridate}` package.

`lubridate::(floor_date)` takes a date-time object and rounds it down to the nearest boundary of the specified unit.

KEY POINT

The `unit` argument of `lubridate::(floor_date)` can specify either one of the following as character strings. (Second, minute, hour, day, week, month, bimonth, quarter, season, halfyear and year.)

For example, let's apply `floor_date()` to today's date. `floor_date()` takes the current date provided by the function `lubridate::today()` and rounds it down to the first day of the particular month as shown below.

```
lubridate::today() #gives the current date in yyyy-mm-dd format
```

```
## [1] "2022-06-22"
```

```
lubridate::floor_date(lubridate::today(), unit = "months")
```

```
## [1] "2022-06-01"
```

What would be the result if the function `lubridate::floor_date(x, unit= "months")` is applied to Valentine's day 2022? In other words, round 2022-02-14 down to the first day of the particular month in the year 2022.

```
q1 <- "YOUR ANSWER HERE"
.check_q1()
```

PRACTICE (in RMD)

```
## Correct! You are wicked!
##   1   2
```

```
.hint_q1()
```

```
## In May 2014, there were 57 Ebola cases, 243 Ebola cases in
June 2014
##   and 351 Ebola cases in July 2014
```

A.

```
## [1] "2021-02-01"
```

B.


```
## [1] "2022-02-01"
```

C.

```
## [1] "2022-03-01"
```

2.1.1 Monthly cases

Let us get the number of total Ebola cases per month.

Assumptions

To answer this question, we assume that each individual `date_of_onset` entry represents a unique case of Ebola.

Thus, to get the total unique cases per month, we would use the `{dplyr}` function `mutate()` and the `lubridate::floor_date()` to modify the `date of onset` column and separate the data by months.

That is what this section of code does.

```
dplyr::mutate(date_of_onset = lubridate::floor_date(date_of_onset, unit =
        "months"))
```

Then, the following section of code `count(date_of_onset)` counts the number of unique instances of `date_of_onset` grouped by months.

To ensure that, all possible dates are included in the new column month_onset, use the `complete()` function from the `{tidyr}` package to create a sequence of dates and the argument `fill=list(n=0)`sets to zero (0) any dates that did not have corresponding inputs (cases). Run `?complete` for more details.

Without this step, any graph or table you create may not display time units (that is months) with zero reported cases.

The code below follows the sequence described to get the total monthly cases

```
total_monthly_cases <- ebola_data%>%
  mutate(month_onset = lubridate::floor_date(date_of_onset, unit =
        "months"))%>%
  # new column, 1st of month of onset
  count(month_onset) %>% # count cases by month
  tidyr::complete(
    month_onset = seq.Date(
      min(month_onset, na.rm=T),# include all months with no cases reported
      max(month_onset, na.rm=T),
      by="month"),
    fill = list(n = 0))
```

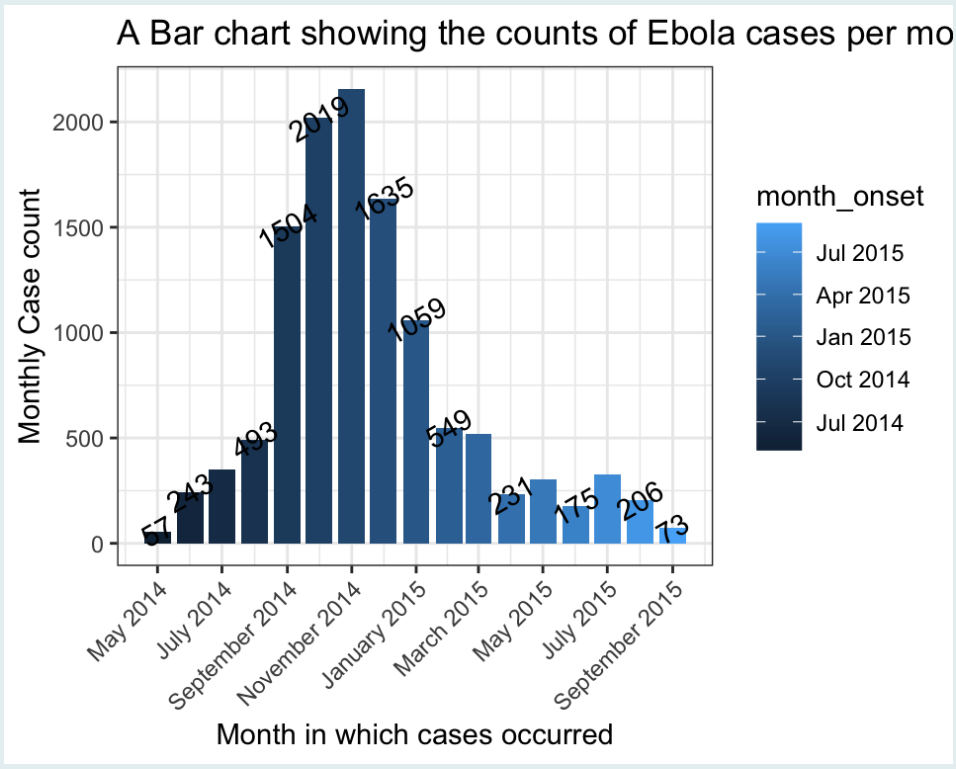
REMINDER

We are using mutate, group_by, count and filter functions from the {dpylr} package (part of the tidyverse packages) for data wrangling and the {ggplot2} package for plotting graphs.

2.1.1.1 Plotting Monthly case totals

Run the following code to plot the monthly Ebola case totals

```
total_monthly_cases%>%
  ggplot(aes(x=month_onset,y=n,fill= month_onset))+geom_col()+
  scale_x_date(date_breaks= "2 month",date_labels = "%B %Y")+
  guides(x = guide_axis(angle = 45))+
  labs(title= "A Bar chart showing the counts of Ebola cases per month")+
  ylab("Monthly Case count")+xlab("Month in which cases occurred")+
  geom_text(aes(label=n),angle=30,check_overlap= T)# to display case counts as
        labels
```



PRO TIP

`guides( x = guide_axis(angle = 45)` is a `{ggplot2}` function that can be used to change the direction of the axis labels on a graph. In the code above we use it to create diagonal axis labels by indicating `angle= 45`.

PRACTICE (in RMD)

Refer to the graph above and find the average number of Ebola cases (mean) for the 3 month period from May 2014 to July 2014. Select the correct answer from the options below.

```
q1 <- "YOUR ANSWER HERE"
.check_q2()
.hint_q2()
```

A.

```
## [1] 217
```

B.

```
## [1] 362.3333
```

C.

```
## [1] 782.6667
```

2.1.2 Daily Cases

Now imagine we were monitoring the cases real-time in 2014 at the very beginning of the epidemic, a case of Ebola has been declared and we need to start looking day by day if we have a growing epidemic on our hands.

Let's follow the code below to plot the daily case rate over May 2014 to see the progression of the spread of Ebola.

```
daily_cases <- ebola_data %>%

  count(date_of_onset) %>%      # count number of rows per unique date
  complete(                      # ensure all days appear even if no cases
    date_of_onset = seq.Date(    # re-define date column as daily sequence of
        dates
      from = min(date_of_onset, na.rm=T),
      to = max(date_of_onset, na.rm=T),
      by = "day"),
    fill = list(n = 0))

daily_cases
```

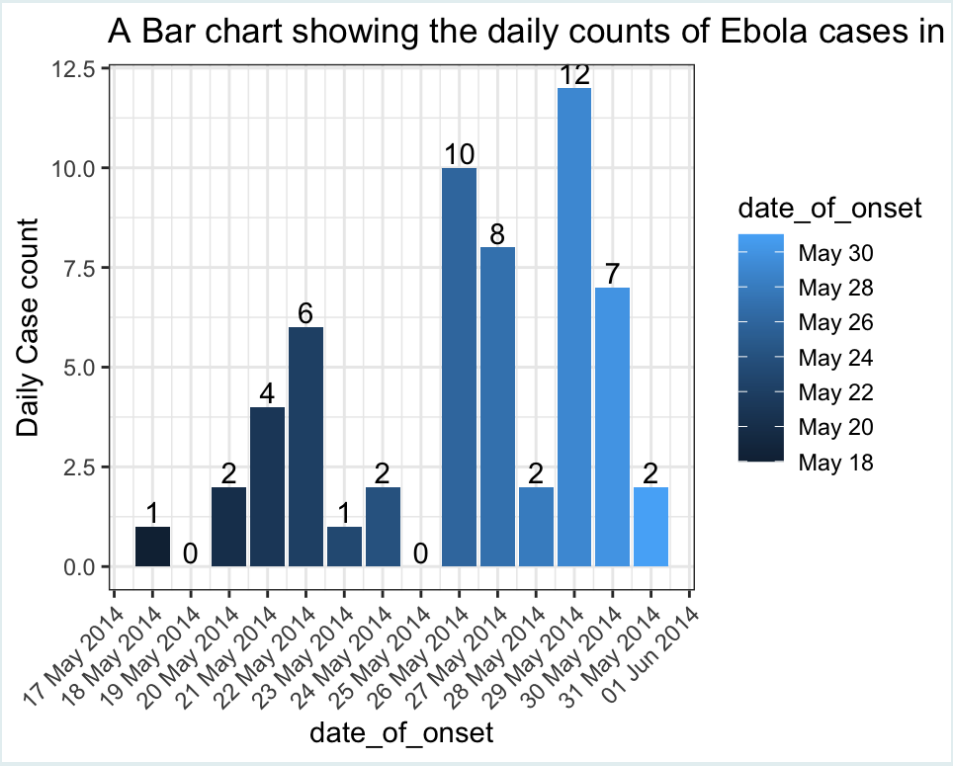
```
## # A tibble: 483 × 2
##    date_of_onset     n
##    <date>        <int>
##  1 2014-05-18        1
##  2 2014-05-19        0
##  3 2014-05-20        2
##  4 2014-05-21        4
##  5 2014-05-22        6
##  6 2014-05-23        1
##  7 2014-05-24        2
##  8 2014-05-25        0
##  9 2014-05-26       10
## 10 2014-05-27        8
## # … with 473 more rows
```

The table above shows a sequence of dates from May 2014 to September 2015 and their associated record of new cases of Ebola. To plot the trend of cases in the first month of the epidemic, we would have to isolate the cases that occurred in May 2014.

Thus, we would use the `lubridate::month()` function to create a new column containing only the month component of the date_of_onset. Also, the `lubridate::year()` will be used to extract the year column.

2.1.2.1 Plotting daily case totals

```
daily_cases%>%
  mutate(month_onset=lubridate::month(date_of_onset, label = TRUE, abbr =
         FALSE),
          year_onset= lubridate::year(date_of_onset))%>%
  filter(month_onset=="May" & year_onset== 2014)%>%
  ggplot(aes(x=date_of_onset,y=n,fill= date_of_onset))+geom_col()+
  scale_x_date(date_breaks= "1 day",date_labels = "%d %b %Y")+
  guides(x = guide_axis(angle = 45))+
  labs(title= "A Bar chart showing the daily counts of Ebola cases in May
         2014")+
  ylab("Daily Case count")+
  geom_text(aes(label=n),check_overlap= T, vjust= -0.2)
```

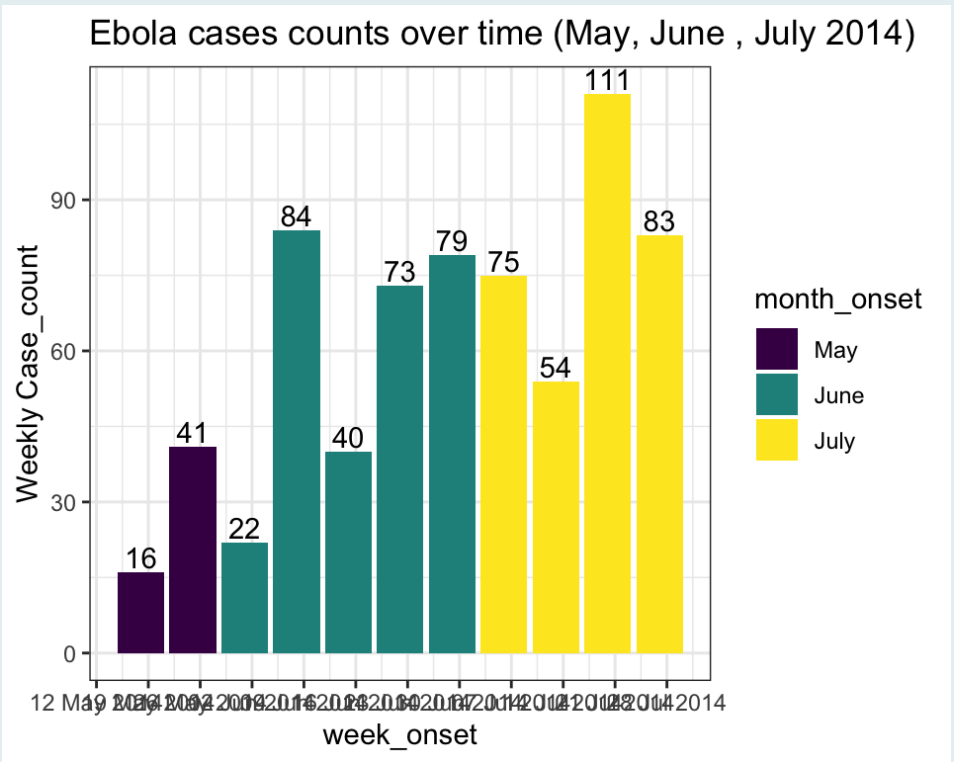


2.1.3 Weekly Case counts

The plot above shows the trend of the first days of the Ebola outbreak in May 2014.

If we wanted to get weekly totals of cases in May and June 2014, we would use the code below.

```
weekly_counts <-
  ebola_data%>%# remove cases missing date_onset
  mutate(week_onset = lubridate::floor_date(date_of_onset, unit = "week")) %>%
        # new column of week of onset
  count(week_onset) %>%  # group data by week and count rows per group
  tidyr::complete(    # ensure all days appear even if no cases
    week_onset = seq.Date(# re-define date column as daily sequence of dates
      from = min(week_onset, na.rm=T),
      to = max(week_onset, na.rm=T),
      by = "week"),
    fill = list(n = 0))

weekly_counts%>%
           mutate(year_onset=lubridate::year(week_onset),
                  month_onset=lubridate::month(week_onset, label= TRUE,
                                               abbr= FALSE))%>%
    filter(month_onset %in% c("May","June","July")&
             year_onset==2014)%>%
    ggplot(aes(x=week_onset, y= n, fill= month_onset)) +geom_col()+
  scale_x_date(date_breaks = "1 week", date_labels = "%d %b %Y")+
  labs(title="Ebola cases counts over time (May, June , July 2014)")+
  ylab("Weekly Case_count")+
  geom_text(aes(label=n),check_overlap= T, vjust= -0.2)
```



We can see that generally there is an increase in case counts with a sharp increase in cases in the week of 9th June, 2014.

Contributors

The following team members contributed to this lesson:

AMA KORANTEMA OWUSU-DARKO

Data Analyst
Passionate about equipping health professionals with data analysis skills to promote evidence based research.

References

Some material in this lesson was adapted from the following sources:

- *Batra, Neale, et al. (2021). The Epidemiologist R Handbook. Chapter 28: GIS Basics*. (2021). Retrieved 01 April 2022, from https://epirhandbook.com/en/gis-basics.html
- *Wickham, Hadley, and Garrett Grolemund."R For Data Science."* 16 Dates and times | R for Data Science, 2017. Accessed May 10, 2022 https://r4ds.had.co.nz/dates-and-times.html#time-zones.

3

Data cleaning: Introduction

3.1 Introduction to the lesson

Data cleaning is one of the important steps in data analysis. Data cleaning is the process of transforming dirty data into reliable data that can be analyzed.

KEY POINT

It involves identifying **inaccurate**, **incomplete**, or **improbable** data and resolving potential data inconsistencies or errors to improve your data quality and subsequently, and overall productivity.

3.2 How do you clean data

PRO TIP

While **every dataset requires different techniques** to clean dirty data, there is a need to address these issues in **a systematic way**.

It's important to conserve as much of the original data as possible while also ensuring that you end up with a clean dataset.

In practice, the cleaning process may focus on finding and resolving data points that don't agree or fit with the rest of the dataset in more obvious ways. In particular, the data cleaning process may include(but is not necessarily limited to):

- Formatting data frame column names
- Deleting all blank rows/columns
- Removing duplicate rows
- Checking for irrelevant Observations
- Converting data types into their appropriate types for analysis
- String normalization

In this chapter, we will go through the process of converting messy data, into clean or reliable data that can be analyzed in R. Multiple packages are available in R to clean the data sets. For this course, we will predominantly make use of functions from the `tidyverse` family of R packages, as well as the `janitor` and `linelist` packages.

SIDE NOTE

This is the flow we will follow for this chapter:

1. We will familiarize ourselves with the data set here in this introduction lesson
2. We will check for structural errors in lesson 1 (cleaning column/variable names) and lesson 2 (removing empty rows/columns)
3. We will check for data irregularities in lesson 3 (deduplicating data entries), in lesson 4 (monitoring types and encodings), and in lesson 5 (managing missing values)

PRO TIP

For the **BEST** data cleaning, you need to know what mess / what errors you are **hunting** for by **EXPLORING** the dataset, then you **CLEAN**.

3.3 The Real Messy Data

In terms of the data, we will use a familiar dataset, the Yaounde dataset, which gives the results from a COVID-19 serological survey conducted in Yaounde, Cameroon in late 2020.

RECAP

It should be noted, however, that the version of the Yaounde dataset we have encountered thus far in previous lessons had already gone through the process of data cleaning.

Now, we will look at the raw data (the original data collected) to learn how to clean a messy data set. In addition, some parts of the Yaounde data set have been altered to further illustrate the data cleaning process.

The full dataset can be obtained from Zenodo, and the paper can be viewed here.

While this is not the type of data we will work with for the rest of this course, the cleaning steps we will perform here are good practice for working with real-world linelists.

SIDE NOTE

For practice, we will also make use of data from a case control study investigating the temporal, spatial and household dynamics of typhoid fever in Kasese district, Uganda.

3.4 Importing and reviewing the data

```
yaounde <-
        read_csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_data.csv'
```

To get a detailed overview of each of the variables in your dataset, there are a number of functions you could use. We recommend the use of the `skim()` function from the `{skimr}` package. This function provides an overview of the data frame and a summary of every column (by class/type).

Specifically, it provides:

- An overview of the rows and columns of the dataframe
- The data type for each variable: `skim_type`
- The number of missing entries (entries per variable that are `NA`): `n_missing`
- The completeness rate for each variable (a number from 0 to 1 corresponding to the ratio: *missing observations / number of observations*): `complete_rate`
- A set of summary statistic; the mean, standard deviation and the five number summary for numerical variables and the frequency and proportions for categorical variables
- Spark histograms and line graphs for the numerical variables

If your "missing" variable is a "0" or a blank space, the `n_missing` and `complete_rate` outputs of `skim()` will not pick up on these missing forms.

```
skimr::skim(yaounde)
```

```
> skimr::skim(yaounde)
── Data Summary ────────────────────────
                           Values
Name                       yaounde
Number of rows             981
Number of columns          36
------------------------
Column type frequency:
  character                20
  logical                  1
  numeric                  15
------------------------
Group variables            None
```

skimr output, part 1

```
── Variable type: character ──────────────────────────────────────────────────────────────────────
   skim_variable           n_missing complete_rate min max empty n_unique whitespace
 1 id ind                          2         0.998  14  20     0      971          0
 2 AGE.CATEGORY                    2         0.998   4   7     0        5          0
 3 SEX                             2         0.998   1   6     0        6          0
 4 EDUCATION                       2         0.998   6  22     0        7          0
 5 OCCUPATION                      2         0.998   5  38     0       28          0
 6 is.smoker                       4         0.996   6  10     0        3          0
 7 is.pregnant                   432         0.560   2  11     0        3          0
 8 is.medicated                    2         0.998   2  11     0        3          0
 9 household with_children         2         0.998  11  13     0        2          0
10 breadwinner                     2         0.998   6  28     0       13          0
11 source of_revenue               2         0.998   5  53     0       20          0
12 has contact_COVID               2         0.998  16  26     0        3          0
13 igg.result                      2         0.998   8   8     0        2          0
14 igm result                      2         0.998   8   8     0        2          0
15 symptoms..                      2         0.998   5  75     0      122          0
16 consultation                  906         0.0765  5  31     0       17          0
17 treatment..combinations       681         0.306   5  80     0       33          0
18 drugsource                    710         0.276   5  28     0        7          0
19 hospitalised                    2         0.998   2  25     0        5          0
20 sequelae                      969         0.0122  5  19     0        4          0

── Variable type: logical ────────────────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate mean count
1 NA                  981             0  NaN ": "
```

skimr output, part 2

```
── Variable type: numeric ────────────────────────────────────────────────────────────────────────
   skim_variable           n_missing complete_rate      mean      sd p0   p25 p50   p75 p100 hist
 1 AGE                             2         0.998   29.0     17.3     5  15    26  39     79 ▆▇▅▂▁
 2 weight kg                       2         0.998   64.5     23.1    14  51    66  78    162 ▃▇▇▂▁
 3 height cm                       2         0.998  159       18.7    54 154   164 170    196 ▁▂▅▇▂
 4 respiration frequency.        963         0.0183  21.6      6.76   15  16.2  20  24.5  37 ▇▃▂▁▁
 5 is drug_parac                 681         0.306    0.557    0.498   0   0     1   1     1 ▆▁▁▁▇
 6 is drug_antibio               681         0.306    0.27     0.445   0   0     0   1     1 ▇▁▁▁▃
 7 is drug_hydrocortisone        681         0.306    0.05     0.218   0   0     0   0     1 ▇▁▁▁▁
 8 is drug_other_anti_inflam     681         0.306    0.0633   0.244   0   0     0   0     1 ▇▁▁▁▁
 9 is drug_antiviral             681         0.306    0        0       0   0     0   0     0 ▁▁▇▁▁
10 is drug_chloro                681         0.306    0.00667  0.0815  0   0     0   0     1 ▇▁▁▁▁
11 is drug_tradn                 681         0.306    0.37     0.484   0   0     0   1     1 ▇▁▁▁▅
12 is drug_oxygen                681         0.306    0.00667  0.0815  0   0     0   0     1 ▇▁▁▁▁
13 is drug_other                 681         0.306    0.223    0.417   0   0     0   0     1 ▇▁▁▁▂
14 is drug_no_resp               681         0.306    0.0167   0.128   0   0     0   0     1 ▇▁▁▁▁
15 is drug_none                  681         0.306    0.0933   0.291   0   0     0   0     1 ▇▁▁▁▁
```

skimr output, part 3

You can also specify the specific variables to be included, similar to `dplyr::select()`

```
skimr::skim(yaounde,SEX,AGE)
```

Table 3.1: Data summary

Name	yaounde
Number of rows	981
Number of columns	36
_____	

Column type frequency:	
character	1
numeric	1
————————	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
SEX	2	1	1	6	0	6	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
AGE	2	1	29.01	17.33	5	15	26	39	79	▇▆▃▂▁

The format of the results are a single wide data frame combining the summary results, with some additional attributes and two metadata columns:

- `skim_variable`: name of the original variable
- `skim_type`: class of the variable

We can also use some `dplyr` verbs like `select` on skim output. We only need to specify the additional attributes mentioned above.

1. Use `skim()` to obtain a detailed overview of the Ugandan `typhoid` dataset.
2. Use `skim()` to obtain a detailed overview of the Ugandan `typhoid` dataset. However, this time, include only the variables `Age` and `Levelofeducation`.

To just get information on a variable without displaying the type of variable

```
skim(yaounde) %>%
  select(skim_type) %>%
  count(skim_type)
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
```

```
## ✖ Column `skim_type` doesn't exist.
```

Overall, the output displayed above highlights a few key features about our data frame

- There are 981 rows and 36 columns.
- There are 20 character variables, 15 numeric variables and 1 logical variable.

In addition, looking at the output from the `skim()` function immediately brings to light some potential issues in the data that may need to be addressed before the data are used in an analysis.

We can identify the following issues with our data :

- there is an empty column in the data (the logical column, `NA` has a `complete_rate` of 0)
- names of variables are unclear/unclean (`is drug_parac` has a whitespace in its name, `treatment..combinations` has a special character, `..`, etc.)
- several variables are heavily incomplete (such as `sequelae`, `consultation`)
- distributions are skewed for certain variables (such as `AGE`, `height cm`)
- etc...

- What other potential issues from the `skim()` output may need to be addressed before the data are used in an analysis?

Contributors

The following team members contributed to this lesson:

LAURE VANCAUWENBERGHE
Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Batra, Neale, et al. The Epidemiologist R Handbook. 2021.*Cleaning data and core functions*. https://epirhandbook.com/en/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions
- Waring E, Quinn M, McNamara A, Arino de la Rubia E, Zhu H, Ellis S (2022). skimr: Compact and Flexible Summaries of Data. https://docs.ropensci.org/skimr/ (website), https://github.com/ropensci/skimr/.

4

Data cleaning: tidying column names

4.1 Learning objectives

1. You can use `janitor::clean_names()` to column names automatically.
2. You can use `rename()` and `rename_with()` to clean column names manually.

4.2 Intro to the lesson

In R, column names are the "header" or "top" value of a column. They are used to refer to columns in the code, and serve as a default label in figures. They should have "clean", standardized syntax so that we can work with them, and so that our code can be readable to other coders.

Ideally, column names:

- should be short
- should have no spaces or periods(space and periods should be replaced by underscore "_")
- should have no unusual characters(&, #, <, >)
- should have a similar style

In this lesson, we will explore how to clean column names manually and automatically in R.

As a reminder, the data we are cleaning is the data from the COVID-19 serological survey conducted in Yaounde, Cameroon.

```
yaounde <-
        read_csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_data.csv'
```

4.3 Columns of our Data

We can use the `names()` function from base R. Or return to this chapter's intro to have a look at the output of `skim()`

```
names(yaounde)
```

```
##  [1] "id ind"                    "AGE"
"AGE.CATEGORY"              "SEX"                       "EDUCATION"
##  [6] "OCCUPATION"                "weight kg"                 "height cm"
"is.smoker"                 "is.pregnant"
## [11] "is.medicated"              "household with_children"   "breadwinner"
"source of_revenue"         "has contact_COVID"
## [16] "igg.result"                "igm result"                "symptoms.."
"consultation"              "treatment..combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration frequency."    "is drug_parac"
## [26] "is drug_antibio"           "is drug_hydrocortisone"    "is
drug_other_anti_inflam" "is drug_antiviral"         "is drug_chloro"
## [31] "is drug_tradn"             "is drug_oxygen"            "is
drug_other"             "is drug_no_resp"           "is drug_none"
## [36] "NA"
```

We can see that:

- some names contain spaces
- some names contain special characters such as `..`
- some names are in upper case while some are not

1. Display the names of the `typhoid` dataset.

4.4 Automatically clean column names

4.4.1 `janitor::clean_names()`

A handy function for standardizing column names is the `clean_names()` from the {janitor} package.

The function `clean_names()`:-

KEY POINT

- Converts all names to consist of only underscores, numbers, and letters.
- Parses letter cases and separators to a consistent format. (default is snake_case)
- Handles special characters(&, #, <, >) or accented characters.

```
yaounde %>%
  clean_names() %>%
  names()
```

```
##  [1] "id_ind"                    "age"
"age_category"              "sex"                       "education"
##  [6] "occupation"                "weight_kg"                 "height_cm"
"is_smoker"                 "is_pregnant"
## [11] "is_medicated"              "household_with_children"   "breadwinner"
"source_of_revenue"         "has_contact_covid"
## [16] "igg_result"                "igm_result"                "symptoms"
"consultation"              "treatment_combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration_frequency"     "is_drug_parac"
## [26] "is_drug_antibio"           "is_drug_hydrocortisone"
"is_drug_other_anti_inflam" "is_drug_antiviral"         "is_drug_chloro"
## [31] "is_drug_tradn"             "is_drug_oxygen"
"is_drug_other"             "is_drug_no_resp"           "is_drug_none"
## [36] "na"
```

RECAP

From this output, we can see that:

- upper case variable names were converted to lower case (e.g., `EDUCATION` is now `education`)
- spaces inside the variable names have been converted to underscores (e.g., `id ind` is now `id_ind`)
- periods(`.`) have all been replaced by underscores (e.g., `is.smoker` is now `is_smoker`)

2. Use the `clean_names()` function from {janitor} to clean the variables names `typhoid` dataset. **Do not display the column names**

4.5 Manually cleaning column names

We can also rename columns manually, either as an alternative to the automatic procedure described above or **in addition to the automatic step above**.

We will first illustrate how to manually clean column names as an alternative to the automatic procedure using the functions `rename()` and `rename_with()`.

4.5.1 `rename()` and `rename_with()`

The `rename()` function is simply a way to change variable names. It was covered in our data wrangling chapter.

As a reminder, its syntax is so: `rename(new_name = old_name)`

Here, manually, we could rename `weight kg` to `weight`, `height cm` to `height`, `is.smoker` to `smoker`, `is.pregnant` to `pregnant` and `is.medicated` to `medicated`.

```
yaounde %>% rename(weight = `weight kg`,
                   height = `height cm`,
```

```
                    smoker = is.smoker,
                    pregnant = is.pregnant,
                    medicated = is.medicated) %>%              names()
```

But this would make a long list if we have a dataset with 300 columns. Let's see how to do so with `rename_with()`.

Instead of changing each column individually, `rename_with` is a handy shortcut to do the same modification to all (or a collection of) column names.

The arguments of `rename_with` are: a data frame

- a function to apply to each column name
- a selection of column to rename (by default all columns)

Say, for example, we want all the column names to be in lower case, we can either change all the columns in upper case to lower case, 1 at a time (which would be a time consuming exercise), or we can use the rename_with function;

```
yaounde %>%
  rename_with(tolower) %>%
  names()
```

```
##  [1] "id ind"                    "age"
"age.category"              "sex"                       "education"
##  [6] "occupation"                "weight kg"                 "height cm"
"is.smoker"                 "is.pregnant"
## [11] "is.medicated"              "household with_children"   "breadwinner"
"source of_revenue"         "has contact_covid"
## [16] "igg.result"                "igm result"                "symptoms.."
"consultation"              "treatment..combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration frequency."    "is drug_parac"
## [26] "is drug_antibio"           "is drug_hydrocortisone"    "is
drug_other_anti_inflam" "is drug_antiviral"         "is drug_chloro"
## [31] "is drug_tradn"             "is drug_oxygen"            "is
drug_other"             "is drug_no_resp"           "is drug_none"
## [36] "na"
```

Here we see that all columns such as `AGE` and `SEX` etc. have been transformed to lower case.

3. Manually rename the following column names in the `typhoid` dataset; `CaseorControl`and `Levelofeducation` to `case_control`, `education_level`, respectively.

4. Convert all the column names in the `typhoid` dataset to lower case

str_replace

The `rename_with` function can also be used for more complex operations, such as, modifying column names with unwanted characters (whitespaces, special characters).

We use a new function called `str_replace` to replace the whitespaces from our column names with an underscore.

The function allows us to replace matched patterns in a string. The arguments are:

- a pattern such as `"is."` or `"-"`
- a replacement pattern

Let's build on what we did before. We corrected for upper cases, now let's correct whitespaces.

```
yaounde %>%
  rename_with(tolower) %>%
  rename_with(str_replace, pattern = " ",replacement = "_") %>%
  names()
```

```
##  [1] "id_ind"                    "age"
"age.category"                "sex"                       "education"
##  [6] "occupation"                "weight_kg"                 "height_cm"
"is.smoker"                   "is.pregnant"
## [11] "is.medicated"              "household_with_children"   "breadwinner"
"source_of_revenue"           "has_contact_covid"
## [16] "igg.result"                "igm_result"                "symptoms.."
"consultation"                "treatment..combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration_frequency."      "is_drug_parac"
## [26] "is_drug_antibio"           "is_drug_hydrocortisone"
"is_drug_other_anti_inflam"   "is_drug_antiviral"         "is_drug_chloro"
## [31] "is_drug_tradn"             "is_drug_oxygen"
"is_drug_other"               "is_drug_no_resp"           "is_drug_none"
## [36] "na"
```

WATCH OUT

To replace special characters like the period (`.`), we adjust the syntax slightly by adding `\\` before the special character.

```
yaounde %>%
  #correcting for upper cases
  rename_with(tolower) %>%
  #correcting for white spaces
  rename_with(str_replace, pattern = " ",replacement = "_") %>%
  #correcting for periods
  rename_with(str_replace, pattern = "\\.",replacement = "_") %>%
  names()
```

```
##  [1] "id_ind"                    "age"
"age_category"                "sex"                       "education"
##  [6] "occupation"                "weight_kg"                 "height_cm"
"is_smoker"                   "is_pregnant"
## [11] "is_medicated"              "household_with_children"   "breadwinner"
"source_of_revenue"           "has_contact_covid"
## [16] "igg_result"                "igm_result"                "symptoms_."
"consultation"                "treatment_.combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration_frequency_"      "is_drug_parac"
## [26] "is_drug_antibio"           "is_drug_hydrocortisone"
"is_drug_other_anti_inflam"   "is_drug_antiviral"         "is_drug_chloro"
## [31] "is_drug_tradn"             "is_drug_oxygen"
"is_drug_other"               "is_drug_no_resp"           "is_drug_none"
## [36] "na"
```

WATCH OUT

want to replace all occurrences of `.` we can use the function `str_replace_all`.

Here we see that `treatment..combinations` has become `treatment._combinations`: still not ideal as a column name.

```
yaounde %>%
  #correcting for upper cases
  rename_with(tolower) %>%
  #correcting for ALL white spaces
  rename_with(str_replace_all, pattern = " ",replacement = "_") %>%
  #correcting for ALL periods
  rename_with(str_replace_all, pattern = "\\.",replacement = "_") %>%
  names()
```

```
##  [1] "id_ind"                    "age"
"age_category"              "sex"                       "education"
##  [6] "occupation"                "weight_kg"                 "height_cm"
"is_smoker"                 "is_pregnant"
## [11] "is_medicated"              "household_with_children"   "breadwinner"
"source_of_revenue"         "has_contact_covid"
## [16] "igg_result"                "igm_result"                "symptoms__"
"consultation"              "treatment__combinations"
## [21] "drugsource"                "hospitalised"              "sequelae"
"respiration_frequency_"    "is_drug_parac"
## [26] "is_drug_antibio"           "is_drug_hydrocortisone"
"is_drug_other_anti_inflam" "is_drug_antiviral"         "is_drug_chloro"
## [31] "is_drug_tradn"             "is_drug_oxygen"
"is_drug_other"             "is_drug_no_resp"           "is_drug_none"
## [36] "na"
```

But even with this, we still have weird column names such as `symptoms__` or `treatment__combinations`. These names are cleaned automatically by {janitor}'s `clean_names()` to `symptoms` and `treatment_combinations`: remember to use this function first! **It will save you a lot of trouble/manipulations.**

5

Automatic then manual cleaning of column names

Evidently, manually cleaning column names can be a cumbersome task, particularly when cleaning a large dataset with many variables. A combination of the automatic and manual procedure is more desirable as it not only saves on time, but can help make our column names more readable. We can start with the automatic clean-up, then check there are no weird column names remaining (which would need manual cleanup).

Here is an example of combining automatic and manual cleaning: a more appropriate cleanup would be to standardise the column names, first, and then remove the prefix `is_` from all columns.

```
yaounde %>%
  # standardize column name syntax
  clean_names() %>%
  # manually re-name columns
  rename_with(str_replace_all, pattern = "is_",replacement = "")
```

Now let's save this first step of cleaning as a dataset.

PRO TIP

It is good practice to save your dataset regularly in different steps of cleaning. In case you should find later on that your cleaning generated a data manipulation error and need to go back through your cleaning steps !

```
yaounde_cleaned_column_names <-
  yaounde %>%
  # standardize column name syntax
  clean_names() %>%
  # manually re-name columns
  rename_with(str_replace_all, pattern = "is_",replacement = "")

write_csv(yaounde_cleaned_column_names,

          here::here('ch02_data_cleaning_pipeline/data/yaounde_data_clean_names.cs
```

PRACTICE

5. Standardize the column names in the `typhoid` dataset then;

- replace `or_` with `_`
- replace `of` with `_`
- rename variables `below10years` `n1119years` `n2035years` `n3644years`, `n4565years` `above65years` to `num_below_10_yrs` `num_11_19_yrs` `num_20_35_yrs` `num_36_44_yrs`, `num_45_65_yrs` `num_above_65_yrs`

5.1 Wrapping up

In this lesson, we have been able to automatically and manually clean column names. This is only the first step to fixing the structural errors in our data. The next step would be removing any empty rows or columns from the data. We will be tackling this in the next lesson.

Contributors

The following team members contributed to this lesson:

LAURE VANCAUWENBERGHE
Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Horst, A. (2021). *Dplyr-learnr*. https://github.com/allisonhorst/dplyr-learnr (Original work published 2020)
-

Batra, Neale, et al. The Epidemiologist R Handbook. 2021. *Cleaning data and core functions*. https://epirhandbook.com/en/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions

Artwork was adapted from:

- Horst, A. (2021). *R & stats illustrations by Allison Horst*. https://github.com/allisonhorst/stats-illustrations (Original work published 2018)

6

Data cleaning: removing empty rows and columns

6.1 Learning objectives

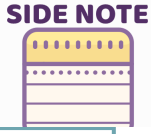
1. You will use `janitor::remove_empty()`, to remove empty rows and columns.
2. You will learn when to remove empty rows and columns and why it is meaningful

6.2 Intro to the lesson

An **EMPTY** row/column is one where all values are `NA` values.

There are two clear moments where you will want to check for empty rows or columns:

1. When you load a dataset, you always want to check if it has any empty rows or columns and remove them. The goal is that **every row is a meaningful data point** and that **every column is a meaningful variable**.
2. Another moment where you will want to check for empty rows or empty columns will be after wrangling the data.



A quick example: if you filter to keep only the children of the dataset, then maybe some of these children will have missing records. With their

SIDE NOTE 

age information they are not empty rows INITALLY but if AFTER FILTERING for the age variable, then they are empty records and should not be considered for the subsequent analysis.

In this lesson, we will explore how to remove empty rows and columns in R.

6.3 Our Data

RECAP 

We are cleaning the data from the COVID-19 serological survey conducted in Yaounde, Cameroon.

We will use the version of the dataset with standardized column names.

```
yaounde <-
        read.csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_data_clear
```

The first step to removing the empty rows and columns is to identify these empty columns and rows.

6.4 Exploring emptiness

We will again use the `inspect_na()` function from the package {inspectdf} to identify empty columns.

```
inspectdf::inspect_na(yaounde)
```

```
## # A tibble: 36 × 3
##    col_name                 cnt  pcnt
##    <chr>                  <int> <dbl>
##  1 na                       981 100
##  2 sequelae                 969  98.8
##  3 respiration_frequency    963  98.2
##  4 consultation             906  92.4
##  5 drugsource               710  72.4
##  6 treatment_combinations   681  69.4
```

```
##  9 drug_hydrocortisone      681  69.4
## 10 drug_other_anti_inflam   681  69.4
## # … with 26 more rows
```

From the output we see that the `pcnt` indicates 100% emptiness (i.e. `NA` values) for this `na` column: there is a `NA` values in every row.

We also see that `sequelae`, `respiration_frequency`, `consultation` are respectively 98.8%, 98.2%, 92.4% incomplete (filled with `NA`). **These variables are unusable for further analysis.**

There was an issue during data collection, and it cannot be undone at the programming level.

The less data you have, the more you have to leave aside variables which are not documented correctly.

1. Identify the empty columns in the `typhoid` dataset.

6.5 Remove empty columns

In order to remove empty columns from the data frame, we will use the `remove_empty()` function from the {`janitor`} package. This function removes all columns from a data frame that are composed entirely of `NA` values.

The syntax is `remove_empty(data,"cols")`, where data is a data frame and `"cols"` specifies that we are removing empty columns.

We will apply the function on the `yaounde` dataset and remove the empty column identified earlier.

```
ncol(yaounde)
```

```
## [1] 36
```

```
yaounde <- yaounde %>%
  remove_empty("cols")

ncol(yaounde)
```

```
## [1] 35
```

We can see that the column named `na` has been removed from the data.

2. Remove the empty columns from the `typhoid` dataset.

:::

6.6 Remove empty rows

While it is relatively easy to identify empty columns from the `skim()` output, its not as easy to do so for empty rows. Fortunately, the `remove_empty()` also works if there are empty rows in the data. The only change in the syntax is specifying `"rows"`instead of `"cols"`.

```
nrow(yaounde)
```

```
## [1] 981
```

```
yaounde <- yaounde %>%
  remove_empty("rows")

nrow(yaounde)
```

```
## [1] 979
```

The number of rows has gone from 981 to 979 suggesting there were empty rows in the data that have been removed.

The `remove_empty()` function can be used to remove empty rows and columns simultaneously. This can be done by adding the arguments `which = c("rows", "cols")`. Also, not specifying this argument will lead to R removing both.

3. Remove both the empty rows and columns from the `typhoid` dataset.

Data is **collected by HUMANS** which means that, in order to indicate an uncollected data, a human might input all `0` (zeros) or `"NA"` (`NA` but as a string)

CAREFUL: `remove_empty()` will not be able to identify these rows/columns as "empty" !

A good practice to identify these could be to look at each column with `unique(column_name)` from base R to see of which unique values it is composed.

Let's save our even cleaner dataset !

```
yaounde_no_empty <- yaounde %>%
  remove_empty(which = c("rows","cols"))

write_csv(yaounde_no_empty,

          here::here('ch02_data_cleaning_pipeline/data/yaounde_data_clean_structu
```

6.7 Wrapping up

In this lesson, we have been able to remove the empty rows and columns from our data. The next step in the data cleaning pipeline would be to check for irregularities in the data. In the next lesson, we will start with deduplicating data entries.

Contributors

The following team members contributed to this lesson:

A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Batra, Neale, et al. The Epidemiologist R Handbook. 2021. *Cleaning data and core functions*. https://epirhandbook.com/en/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions
- Sam Firke, Bill Denney , Chris Haid , Ryan Knight, Malte Grosser , Jonathan Zadra (2021). janitor: Simple Tools for Examining and Cleaning Dirty Data. R package version 2.1.0. https://cran.r-project.org/web/packages/janitor/vignettes/janitor.html

7

Data cleaning: deduplication

7.1 Learning objectives

1. *Exploring duplicates* with `duplicated()` and `get_dupes()` from the {`janitor`} package, to identify rows which values appear more than once.
2. *Removing duplicates* using `unique()` and `distinct()` from the {`dplyr`} package, to remove duplicate rows from a dataset.

7.2 Intro to the lesson

Very often in your datasets there are situations where you have duplicated values of data, when one row has the same values as some other row. This often occurs when you combine data from multiple sources, or have received multiple survey responses.

Duplicated values can lead you to make incorrect conclusions by leading you to believe that some observations are more common than they really are.

As such, it is therefore necessary to identify and remove any duplicate values from your data in order to have well-balanced results. In this lesson, we will explore how to identify and remove duplicated values in R.

7.3 Our data

The data we are cleaning is the data from the COVID-19 serological survey conducted in Yaounde, Cameroon.

We will use the version of the dataset with:

- standardized column names
- with no empty columns and rows.

For this lesson, we will also make use of some vectors to illustrate the concepts.

```
#Data frame
yaounde <-
        read.csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_data_clea

# Vectors
ages <- c(11, 21, 46, 21, 19, 18, 19)
```

7.4 Exploring duplicates

7.4.1 `duplicated()`

The `duplicated()` function from base R defines which items of a vector or data frame are duplicates.

The output specifies the position of duplicate elements (rows) in the vector(data frame). If the element(row) is duplicated, the function returns `TRUE`. The first time a value appears, it will return `FALSE` (not a duplicate), and subsequent times that value appears it will return `TRUE`.

The syntax is `duplicated(x)`, where x can be a vector or a data frame.

We will apply the function on the `ages` vector

```
duplicated(ages)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
```

We can see that the fourth and seventh elements of the `ages` vector are both duplicates.

WATCH OUT

The output above only returns the positions of the duplicates and does not specify which elements are being duplicated.

We will now apply the same function on the `yaounde` dataset.

```
#Let's look at the first 100 rows
duplicated(yaounde) %>% head(100)
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [24] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [47] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [70] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [93] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

From the output, we can tell that the 13th and the 28th row are both duplicates: an identical row has been found beforehand in the dataset.

1. Identify the elements that are duplicates from the `typhoid` dataset. Display only the first 50 entries.

It is, however, quite clear that applying the `duplicated()` function on larger vectors or data frames does not provide easily discernible output. A more useful way of using the `duplicated()` function is **extracting the duplicated elements**.

```
ages[duplicated(ages)]
```

```
## [1] 21 19
```

The output shows that the ages 19 and 21 are duplicates.

Using the same syntax, we can also extract the duplicate rows from the yaounde dataset.

```
yaounde[duplicated(yaounde),]
```

```
##                   id_ind age age_category    sex    education
occupation weight_kg height_cm     smoker pregnant   medicated
## 13   BRIQUETERIE_003_0005  16      15 - 29 Female   Secondary1
```

```
Student        70       164 Non-smoker       NO          NO
## 28  BRIQUETERIE_007_0002  15      15 - 29   Male   Secondary1
Student        51       173 Non-smoker     <NA>          NO
## 244    CARRIERE_034_0005  52      45 - 64 Female No response1     No
response        74       159     Smoker       NO          NO
## 306    CARRIERE_052_0006  17      15 - 29   Male   Secondary1
Student        66       175 Non-smoker     <NA>          NO
## 428     EKOUDOU_002_0005  44      30 - 44 Female   Secondary1      Home-
maker         78       172 Non-smoker       NO          NO
## 698      MOKOLO_013_0002  25      15 - 29 Female   Secondary1      Home-
maker         80       175 Non-smoker      YES          NO
## 722      MOKOLO_021_0002  46      45 - 64   Male   Secondary1 Informal
worker        122       173 Non-smoker     <NA> NO RESPONSE
## 979 TSINGAOLIGA_005_0011   9       May-14   Male     Primary1
Student        24       135 Non-smoker     <NA> NO RESPONSE
##     household_with_children breadwinner
source_of_revenue          has_contact_covid igg_result igm_result
## 13            WITH CHILDREN NO RESPONSE IRREGULAR, FORMAL--IRREGULAR,
COMMERCE           NO COVID CONTACT   Positive   Positive
## 28            WITH CHILDREN      FATHER                         REGULAR,
FORMAL           NO COVID CONTACT   Negative   Negative
## 244           WITH CHILDREN NO RESPONSE                    IRREGULAR,
COMMERCE           NO COVID CONTACT   Negative   Negative
## 306           WITH CHILDREN      MOTHER                     IRREGULAR,
COMMERCE           NO COVID CONTACT   Negative   Negative
## 428           WITH CHILDREN      FATHER                       REGULAR,
COMMERCE UNSURE ABOUT COVID CONTACT   Positive   Negative
## 698           WITH CHILDREN      FATHER                     IRREGULAR,
INFORMAL           NO COVID CONTACT   Negative   Negative
## 722           WITH CHILDREN      MOTHER   REGULAR, FORMAL--IRREGULAR,
INFORMAL UNSURE ABOUT COVID CONTACT   Negative   Negative
## 979           WITH CHILDREN NO RESPONSE
OTHER          NO COVID CONTACT   Negative   Positive
##                          symptoms           consultation
treatment_combinations
## 13  Rhinitis--Sneezing--Headache                   <NA>
Paracetamol
## 28                          Fever         Private clinic
Paracetamol--Hydroxychloroquine
## 244                 No symptoms                   <NA>
<NA>
## 306     Fever--Headache--Fatigue Private clinic--Doctor Paracetamol--
Antibiotics--Hydrocortisone--Other anti-inflamm.--Traditional meds.
## 428    Fever--Rhinitis--Headache                   <NA>
Paracetamol--Antibiotics
## 698                 No symptoms                   <NA>
<NA>
## 722              Cough--Sneezing                   <NA>
Paracetamol
## 979                 No symptoms                   <NA>
<NA>
##                   drugsource              hospitalised sequelae
respiration_frequency drug_parac drug_antibio drug_hydrocortisone
## 13          Self or familial                          No     <NA>
NA          1            0                   0
## 28               Pharmacist Yes, but not COVID-linked     <NA>
```

```
NA          1            0                    0
## 244                      <NA> Yes, but not COVID-linked     <NA>
NA         NA           NA                   NA
## 306 Doctor--Self or familial                         No     <NA>
NA          1            1                    1
## 428         Self or familial                         No     <NA>
NA          1            1                    0
## 698                      <NA>                         No     <NA>
NA         NA           NA                   NA
## 722                     Doctor                        No     <NA>
NA          1            0                    0
## 979                      <NA>                          No response     <NA>
NA         NA           NA                   NA
##     drug_other_anti_inflam drug_antiviral drug_chloro drug_tradn
drug_oxygen drug_other drug_no_resp drug_none
## 13                        0              0           0           0
0          0            0          0
## 28                        0              0           1           0
0          0            0          0
## 244                      NA             NA          NA          NA
NA         NA           NA         NA
## 306                       1              0           0           1
0          0            0          0
## 428                       0              0           0           0
0          0            0          0
## 698                      NA             NA          NA          NA
NA         NA           NA         NA
## 722                       0              0           0           0
0          0            0          0
## 979                      NA             NA          NA          NA
NA         NA           NA         NA
```

The output shows that there are 8 rows that are duplicates.

Note the slight difference in syntax for the data frame (there is an additional comma in the square brackets). This is necessary because the dimensions of the dataset are `rows x columns` and the `duplicate()` function is intended for the rows. In this manner we select duplicate rows and all columns.

7.4.2 `get_dupes()`

An alternative to the `duplicated()` function from base R to quickly review rows that have duplicates is the `get_dupes()` function from the {janitor} package.

The syntax is `get_dupes(x)`, where x is **a dataframe**.

```
yaounde %>%
  get_dupes()
```

```
## No variable names specified - using all columns.
```

```
##                  id_ind age age_category    sex    education
occupation weight_kg height_cm     smoker pregnant   medicated
## 1  BRIQUETERIE_003_0005  16      15 - 29 Female   Secondary1
Student         70       164 Non-smoker       NO          NO
## 2  BRIQUETERIE_003_0005  16      15 - 29 Female   Secondary1
Student         70       164 Non-smoker       NO          NO
## 3  BRIQUETERIE_007_0002  15      15 - 29   Male   Secondary1
Student         51       173 Non-smoker     <NA>          NO
## 4  BRIQUETERIE_007_0002  15      15 - 29   Male   Secondary1
Student         51       173 Non-smoker     <NA>          NO
## 5     CARRIERE_034_0005  52      45 - 64 Female No response1     No
response        74       159     Smoker       NO          NO
## 6     CARRIERE_034_0005  52      45 - 64 Female No response1     No
response        74       159     Smoker       NO          NO
## 7     CARRIERE_052_0006  17      15 - 29   Male   Secondary1
Student         66       175 Non-smoker     <NA>          NO
## 8     CARRIERE_052_0006  17      15 - 29   Male   Secondary1
Student         66       175 Non-smoker     <NA>          NO
## 9      EKOUDOU_002_0005  44      30 - 44 Female   Secondary1      Home-
maker           78       172 Non-smoker       NO          NO
## 10     EKOUDOU_002_0005  44      30 - 44 Female   Secondary1      Home-
maker           78       172 Non-smoker       NO          NO
## 11      MOKOLO_013_0002  25      15 - 29 Female   Secondary1      Home-
maker           80       175 Non-smoker      YES          NO
## 12      MOKOLO_013_0002  25      15 - 29 Female   Secondary1      Home-
maker           80       175 Non-smoker      YES          NO
## 13      MOKOLO_021_0002  46      45 - 64   Male   Secondary1 Informal
worker         122       173 Non-smoker     <NA> NO RESPONSE
## 14      MOKOLO_021_0002  46      45 - 64   Male   Secondary1 Informal
worker         122       173 Non-smoker     <NA> NO RESPONSE
## 15 TSINGAOLIGA_005_0011   9       May-14   Male     Primary1
Student         24       135 Non-smoker     <NA> NO RESPONSE
## 16 TSINGAOLIGA_005_0011   9       May-14   Male     Primary1
Student         24       135 Non-smoker     <NA> NO RESPONSE
##    household_with_children breadwinner
source_of_revenue          has_contact_covid igg_result igm_result
## 1            WITH CHILDREN NO RESPONSE IRREGULAR, FORMAL--IRREGULAR,
COMMERCE           NO COVID CONTACT   Positive   Positive
## 2            WITH CHILDREN NO RESPONSE IRREGULAR, FORMAL--IRREGULAR,
COMMERCE           NO COVID CONTACT   Positive   Positive
## 3            WITH CHILDREN      FATHER                         REGULAR,
FORMAL           NO COVID CONTACT   Negative   Negative
## 4            WITH CHILDREN      FATHER                         REGULAR,
FORMAL           NO COVID CONTACT   Negative   Negative
## 5            WITH CHILDREN NO RESPONSE                     IRREGULAR,
COMMERCE           NO COVID CONTACT   Negative   Negative
## 6            WITH CHILDREN NO RESPONSE                     IRREGULAR,
```

```
COMMERCE           NO COVID CONTACT   Negative   Negative
## 7            WITH CHILDREN      MOTHER                     IRREGULAR,
COMMERCE           NO COVID CONTACT   Negative   Negative
## 8            WITH CHILDREN      MOTHER                     IRREGULAR,
COMMERCE           NO COVID CONTACT   Negative   Negative
## 9            WITH CHILDREN      FATHER                       REGULAR,
COMMERCE UNSURE ABOUT COVID CONTACT   Positive   Negative
## 10           WITH CHILDREN      FATHER                       REGULAR,
COMMERCE UNSURE ABOUT COVID CONTACT   Positive   Negative
## 11           WITH CHILDREN      FATHER                     IRREGULAR,
INFORMAL           NO COVID CONTACT   Negative   Negative
## 12           WITH CHILDREN      FATHER                     IRREGULAR,
INFORMAL           NO COVID CONTACT   Negative   Negative
## 13           WITH CHILDREN      MOTHER   REGULAR, FORMAL--IRREGULAR,
INFORMAL UNSURE ABOUT COVID CONTACT   Negative   Negative
## 14           WITH CHILDREN      MOTHER   REGULAR, FORMAL--IRREGULAR,
INFORMAL UNSURE ABOUT COVID CONTACT   Negative   Negative
## 15           WITH CHILDREN NO RESPONSE
OTHER              NO COVID CONTACT   Negative   Positive
## 16           WITH CHILDREN NO RESPONSE
OTHER              NO COVID CONTACT   Negative   Positive
##                         symptoms           consultation
treatment_combinations
## 1  Rhinitis--Sneezing--Headache                   <NA>
Paracetamol
## 2  Rhinitis--Sneezing--Headache                   <NA>
Paracetamol
## 3                          Fever         Private clinic
Paracetamol--Hydroxychloroquine
## 4                          Fever         Private clinic
Paracetamol--Hydroxychloroquine
## 5                    No symptoms                   <NA>
<NA>
## 6                    No symptoms                   <NA>
<NA>
## 7      Fever--Headache--Fatigue Private clinic--Doctor Paracetamol--
Antibiotics--Hydrocortisone--Other anti-inflamm.--Traditional meds.
## 8      Fever--Headache--Fatigue Private clinic--Doctor Paracetamol--
Antibiotics--Hydrocortisone--Other anti-inflamm.--Traditional meds.
## 9     Fever--Rhinitis--Headache                   <NA>
Paracetamol--Antibiotics
## 10    Fever--Rhinitis--Headache                   <NA>
Paracetamol--Antibiotics
## 11                   No symptoms                   <NA>
<NA>
## 12                   No symptoms                   <NA>
<NA>
## 13               Cough--Sneezing                   <NA>
Paracetamol
## 14               Cough--Sneezing                   <NA>
Paracetamol
## 15                   No symptoms                   <NA>
<NA>
## 16                   No symptoms                   <NA>
<NA>
##                  drugsource              hospitalised sequelae
```

```
respiration_frequency drug_parac drug_antibio drug_hydrocortisone
## 1           Self or familial                        No      <NA>
NA           1             0                    0
## 2           Self or familial                        No      <NA>
NA           1             0                    0
## 3                 Pharmacist Yes, but not COVID-linked      <NA>
NA           1             0                    0
## 4                 Pharmacist Yes, but not COVID-linked      <NA>
NA           1             0                    0
## 5                       <NA> Yes, but not COVID-linked      <NA>
NA          NA            NA                   NA
## 6                       <NA> Yes, but not COVID-linked      <NA>
NA          NA            NA                   NA
## 7   Doctor--Self or familial                        No      <NA>
NA           1             1                    1
## 8   Doctor--Self or familial                        No      <NA>
NA           1             1                    1
## 9           Self or familial                        No      <NA>
NA           1             1                    0
## 10          Self or familial                        No      <NA>
NA           1             1                    0
## 11                      <NA>                        No      <NA>
NA          NA            NA                   NA
## 12                      <NA>                        No      <NA>
NA          NA            NA                   NA
## 13                      Doctor                      No      <NA>
NA           1             0                    0
## 14                      Doctor                      No      <NA>
NA           1             0                    0
## 15                      <NA>                 No response      <NA>
NA          NA            NA                   NA
## 16                      <NA>                 No response      <NA>
NA          NA            NA                   NA
##    drug_other_anti_inflam drug_antiviral drug_chloro drug_tradn
drug_oxygen drug_other drug_no_resp drug_none dupe_count
## 1                       0              0           0           0
0           0             0          0           2
## 2                       0              0           0           0
0           0             0          0           2
## 3                       0              0           1           0
0           0             0          0           2
## 4                       0              0           1           0
0           0             0          0           2
## 5                      NA             NA          NA          NA
NA          NA            NA         NA           2
## 6                      NA             NA          NA          NA
NA          NA            NA         NA           2
## 7                       1              0           0           1
0           0             0          0           2
## 8                       1              0           0           1
0           0             0          0           2
## 9                       0              0           0           0
0           0             0          0           2
## 10                      0              0           0           0
0           0             0          0           2
## 11                     NA             NA          NA          NA
```

```
NA          NA          NA          NA          2
## 12                     NA                  NA          NA          NA
NA          NA          NA          NA          2
## 13                     0                   0           0           0
0           0           0           0           2
## 14                     0                   0           0           0
0           0           0           0           2
## 15                     NA                  NA          NA          NA
NA          NA          NA          NA          2
## 16                     NA                  NA          NA          NA
NA          NA          NA          NA          2
```

The output is made up of 16 rows: there are 2 rows for each pair of duplicates. You can easily see they are duplicates based on this `id_in`.

There is also an additional variable, `dupe_count`, showing the number of rows sharing that combination of duplicated values.

So if there were 5 copies of the same row (i.e. datapoint) then `dupe_count` would be equal to 5.

Careful that if there are many duplicates in a dataset, you should question the rigor of the data collection.

2. Extract the duplicate rows from the `typhoid` dataset.

7.5 Extracting unique elements (Removing duplicates)

7.5.1 `unique()`

The `unique()` base function in R is used to eliminate the duplicate values or the rows present in the vector or data frame.

The `unique()` function works in opposite way of `duplicated()` function in that it returns a vector or data frame with duplicate elements and rows deleted.

The syntax is `unique(x)`, where x can be a vector or a data frame.

Applying the `unique()` function to the ages vector removes the duplicate elements from the vector and returns a vector of unique elements.

```
unique(ages)
```

```
## [1] 11 21 46 19 18
```

It's also possible to apply `unique()` on a data frame, for removing duplicated rows as follows,

```
dim(yaounde) # to get the dimensions of the dataframe
```

```
## [1] 979  35
```

```
yaounde_unique <- unique(yaounde)

dim(yaounde_unique) # to get the dimensions of the dataframe without
        duplicates
```

```
## [1] 971  35
```

Initially, this dataset had 979 rows and 35 columns. Applying the `unique()` function reduces the dimensions of the dataset to 971 rows and 35 columns.

7.5.2 `distinct()`

The `distinct()` is a function of the {`dplyr`} package that can keep unique/distinct rows from **a data frame**. If there are duplicate rows, only the first row is preserved.

To get the unique rows from the data frame, use the following code.

```
yaounde_distinct <- yaounde %>% distinct()

dim(yaounde_distinct)
```

```
## [1] 971  35
```

The output is exactly the same as would be obtained after applying the `unique()` function.

3. Remove the duplicate rows from the `typhoid` dataset. Ensure only unique rows remain in the dataset

So far we have only looked at cases where rows are exactly the same. There are, however, instances when there is a need to remove duplicate rows based on specific columns of a data frame. The `distinct()` function allows for the removal of rows in a data frame based on unique column values or unique combination of columns values. For this reason, `distinct()` is **better for dataframes**.

When finding distinct column values or combination of values, the `.keep_all = TRUE` attribute is used to retain all other variables in the output data frame.

For study designs, it might be useful to keep only unique combinations within a dataset. As an illustration, if we want to get the unique rows for the `sex` and `age` combination

```
yaounde %>%
  distinct(age,sex,.keep_all = TRUE) %>%
  nrow()
```

```
## [1] 142
```

We now have 134 unique rows based on age and sex.

Only the first row of each combination of age and sex is shown in the output

4. Based on `age` and `county`, find the number unique rows in the `typhoid` dataset.

Let's now save our deduplicated dataset using `distinct()` to keep unique patient IDs (this is good practice because all patients should have different patient IDs).

```
yaounde_no_duplicates <- yaounde %>% distinct(id_ind,.keep_all = TRUE)

write_csv(yaounde_no_duplicates,

          here::here('ch02_data_cleaning_pipeline/data/yaounde_data_deduped.csv')
```

7.6 Wrapping up

In this lesson, we learnt how to identify and remove duplicated values from datasets. The next step in the data cleaning pipeline would be fixing inconsistencies and structural errors in the variables.

Contributors

The following team members contributed to this lesson:

LAURE VANCAUWENBERGHE
Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Batra, Neale, et al. The Epidemiologist R Handbook. 2021. *Cleaning data and core functions*. https://epirhandbook.com/en/cleaning-data-and-core-functions.html#cleaning-data-and-core-functions
- Sam Firke, Bill Denney , Chris Haid , Ryan Knight, Malte Grosser , Jonathan Zadra (2021). janitor: Simple Tools for Examining and Cleaning Dirty Data. R package version 2.1.0. https://cran.r-project.org/web/packages/janitor/vignettes/janitor.html

Artwork was adapted from:

- Horst, A. (2021). *R & stats illustrations by Allison Horst*. https://github.com/allisonhorst/stats-illustrations (Original work published 2018)

8

Data cleaning: data transformation

8.1 Learning objectives

1. You will learn how to fix inconsistencies and structural errors in the variables using `recode`, `replace`, and `gsub`
2. You will verify datatypes
3. You will correct datatypes using `across()` to transform multiple columns.

8.2 Intro to the lesson

Data format determines the kind of manipulations and plotting possible with such data.

An important step in data cleaning is **checking the datatypes** of your variables and **fixing any inconsistencies** that would cost time and accuracy of results later on.

A few elements you want to keep in mind as targets:

- you want strings without typos or multiple versions of the same information
- you want numeric variables to "make sense" (no infinity numbers, weird negative numbers etc.)
- you want numbers as numeric variables
- you want variables made of categories as factors

In this lesson, we will explore how to modify existing columns and create new columns. The order of our column cleanings is as follows:

1. Cleaning character type variables and their strings for inconsistencies
2. Cleaning data types to transform all categorical variables into factors
3. With clean data types, we evaluate if our numerical variables are consistent

In this case it is important to have this order because we want to correct some of the character variables, and it is easier to do so while they are still characters. However, many of our numerical variable are factors so applying consistency checks of numerical variables to them would not make sense. So we convert categorical variables to factors before handling our numerical variables.

Much of the content covered in this lesson builds on what has already been covered in Chapter 4, Lesson 3 of the Introduction to Data analysis course. Have a look before starting !

8.3 Our data

The data we are cleaning is the data from the COVID-19 serological survey conducted in Yaounde, Cameroon.

We will use the version of the dataset with:

- standardized column names
- no empty columns or rows
- no duplicates.

```
#data frame
yaounde <-
        read.csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_data_dedup
```

8.4 Dealing with inconsistencies

Recode values

There are often times when you need to correct some inconsistencies in strings that might interfere with data analysis. This includes;

- typos
- capitalization errors.
- misplaced punctuation

These issues can be fixed manually in the raw data source or we can make the change in the cleaning pipeline. The latter is more transparent and reproducible to anyone else seeking to understand or repeat your analysis.

`recode()`

We can use the `recode()` function within the `mutate()` function to change specific values and to reconcile values not spelt the same.

The syntax is `recode(column_name, old_column_value = new_column_value)`

We will have a look at the `age_category` column

```
yaounde %>% count(age_category, name = "Count")
```

```
##   age_category Count
## 1      15 - 29   325
## 2      30 - 44   212
## 3      45 - 64   153
## 4         65 +    40
## 5       May-14   241
```

We see that there are 5 age groups with one of the groups being `May-14`. We will recode this category to `5-14`.

```
yaounde %>%
  mutate(age_category = recode(age_category,`May-14` = "5-14")) %>%
  count(age_category, name = "Count")
```

```
##   age_category Count
## 1      15 - 29   325
## 2      30 - 44   212
## 3      45 - 64   153
## 4         5-14   241
## 5         65 +    40
```

We can also recode more than 1 value in a column. To do this, we will look at the variable `sex`

```
yaounde %>% count(sex, name = "Count")
```

```
##      sex Count
## 1      F     1
## 2 female     1
## 3 Female   547
## 4      M     1
## 5    Mal     1
## 6   Male   420
```

In this variable, there are capitalization errors and typos. We will recode values in the `sex` column so that `F = Female, female = Female, M = Male and Mal = Male`

```
yaounde %>%
  mutate(sex = recode(sex, `F` = "Female",
                           `female` = "Female",
                           `M` = "Male",
                           `Mal` = "Male")) %>%
  count(sex, name = "Count")
```

```
##      sex Count
## 1 Female   549
```

```
## 2   Male   422
```

1. The variable `householdmembers` from the `typhoid` dataset should represent the number of individuals in a household. Display the different values in the variable.

2. There is a value `01-May` in the `householdmembers` variable. Recode this value to `1-5`.

`replace`

The `replace` function can be used to recode values with simple logical criteria . You can also use a logic condition to specify the rows to change. Within the replace function, we have to specify the name of our data object, the value you want to change and the replacement value.

The general syntax is:

```
mutate(col_to_change = replace(col_to_change,
criteria_for_rows, new value)
```

OR

```
mutate(new_name = replace(col_to_change, criteria for
rows, new value).
```

if you wan to create a new column

One common situation to use `replace()` is changing just one value in one row, using an unique row identifier. This can be useful if you have external information indicating anomalies specific to a patient or to an observation and need to do a very specific edit.

More generally, in cleaning, you will use `replace()` to change multiple rows based on some logic criteria.

Conforming to human rights, we will indicate all patients under 18 years of age as having the `occupation` of `Child`. This is how we would do that:

```
yaounde %>%
    mutate(occupation = replace(occupation, age < 18 , "Child"))
```

```
##                  id_ind age age_category    sex   education
occupation weight_kg height_cm     smoker    pregnant medicated
## 1  BRIQUETERIE_000_0001  45      45 - 64 Female  Secondary1 Informal
worker         95        169 Non-smoker           NO       YES
## 2  BRIQUETERIE_000_0002  55      45 - 64    Mal University1 Salaried
worker         96        185  Ex-smoker         <NA>        NO
## 3  BRIQUETERIE_000_0003  23      15 - 29      M University1
Student         74        180     Smoker         <NA>       YES
## 4  BRIQUETERIE_002_0001  20      15 - 29 Female  Secondary1
Student         70        164 Non-smoker           NO        NO
## 5  BRIQUETERIE_002_0002  55      45 - 64 female    Primary1  Trader--
Farmer         67        147 Non-smoker           NO        NO
## 6  BRIQUETERIE_002_0003  17      15 - 29 Female  Secondary1
Child         65        162 Non-smoker           NO        NO
## 7  BRIQUETERIE_002_0004  13       May-14      F  Secondary1
Child         65        150 Non-smoker           NO        NO
## 8  BRIQUETERIE_003_0001  28      15 - 29   Male  Doctorate1
Student         62        173 Non-smoker         <NA>        NO
## 9  BRIQUETERIE_003_0005  16      15 - 29 Female  Secondary1
Child         70        164 Non-smoker           NO        NO
## 10 BRIQUETERIE_003_0002  30      30 - 44   Male  Secondary1
Trader         73        170 Non-smoker         <NA>        NO
## 11 BRIQUETERIE_003_0003  13       May-14 Female  Secondary1
Child         56        153 Non-smoker           NO        NO
## 12 BRIQUETERIE_003_0004  62      45 - 64 Female    Primary1
Unemployed         53        157 Non-smoker NO RESPONSE         NO
## 13 BRIQUETERIE_005_0001  34      30 - 44 Female    Primary1
Trader         78        164 Non-smoker           NO        NO
## 14 BRIQUETERIE_005_0002  38      30 - 44 Female    Primary1       Home-
maker         58        169 Non-smoker           NO        NO
## 15 BRIQUETERIE_005_0003  42      30 - 44   Male  Secondary1
Trader         71        171  Ex-smoker         <NA>        NO
## 16 BRIQUETERIE_005_0004  28      15 - 29 Female    Primary1 Informal
worker         52        155 Non-smoker           NO        NO
## 17 BRIQUETERIE_005_0005  21      15 - 29 Female    Primary1 Informal
worker         56        160 Non-smoker           NO        NO
## 18 BRIQUETERIE_005_0006   8       May-14 Female    Primary1
Child         26        127 Non-smoker           NO        NO
## 19 BRIQUETERIE_005_0007   8       May-14   Male    Primary1
Child         29        120 Non-smoker         <NA>        NO
## 20 BRIQUETERIE_005_0008   7       May-14   Male    Primary1
Child         20        115 Non-smoker         <NA>        NO
## 21 BRIQUETERIE_005_0009   9       May-14   Male    Primary1
Child         28        134 Non-smoker         <NA>        NO
## 22 BRIQUETERIE_007_0002  15      15 - 29   Male  Secondary1
Child         51        173 Non-smoker         <NA>        NO
## 23 BRIQUETERIE_005_0010  31      30 - 44 Female    Primary1
Trader         75        174 Non-smoker           NO        NO
## 24 BRIQUETERIE_005_0011  38      30 - 44   Male University1 Informal
worker         71        175  Ex-smoker         <NA>        NO
## 25 BRIQUETERIE_005_0012   7       May-14 Female    Primary1
```

```
Child        28       131 Non-smoker          NO        NO
## 26 BRIQUETERIE_007_0001  54      45 - 64   Male University1 Salaried
worker        71       180     Smoker        <NA>        NO
## 27 BRIQUETERIE_007_0003  42      30 - 44 Female    Primary1      Home-
maker       101       175 Non-smoker          NO        NO
## 28 BRIQUETERIE_007_0004  18      15 - 29   Male  Secondary1
Student        64       177 Non-smoker        <NA>        NO
##    household_with_children    breadwinner
source_of_revenue          has_contact_covid igg_result igm_result
## 1              NO CHILDREN    NO RESPONSE
OTHER       RECENT COVID CONTACT   Negative   Negative
## 2              NO CHILDREN   OTHER FAMILY            REGULAR, FORMAL--
REGULAR, COMMERCE           NO COVID CONTACT   Positive   Negative
## 3              NO CHILDREN    NO RESPONSE
OTHER           NO COVID CONTACT   Negative   Negative
## 4            WITH CHILDREN         MOTHER
IRREGULAR, COMMERCE          NO COVID CONTACT   Positive   Negative
## 5            WITH CHILDREN         MOTHER REGULAR, INFORMAL--IRREGULAR,
COMMERCE--OTHER           NO COVID CONTACT   Positive   Negative
## 6            WITH CHILDREN    NO RESPONSE
IRREGULAR, COMMERCE          NO COVID CONTACT   Negative   Negative
## 7            WITH CHILDREN    NO RESPONSE
OTHER           NO COVID CONTACT   Positive   Negative
## 8            WITH CHILDREN    NO RESPONSE        IRREGULAR, FORMAL--
IRREGULAR, COMMERCE          NO COVID CONTACT   Negative   Negative
## 9            WITH CHILDREN    NO RESPONSE        IRREGULAR, FORMAL--
IRREGULAR, COMMERCE          NO COVID CONTACT   Positive   Positive
## 10           WITH CHILDREN    NO RESPONSE        IRREGULAR, FORMAL--
IRREGULAR, COMMERCE          NO COVID CONTACT   Negative   Negative
## 11           WITH CHILDREN    NO RESPONSE        IRREGULAR, FORMAL--
IRREGULAR, COMMERCE          NO COVID CONTACT   Positive   Negative
## 12           WITH CHILDREN   OTHER FAMILY
REGULAR, INFORMAL            NO COVID CONTACT   Positive   Negative
## 13           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE            NO COVID CONTACT   Positive   Negative
## 14           WITH CHILDREN         MOTHER
REGULAR, COMMERCE            NO COVID CONTACT   Negative   Negative
## 15           WITH CHILDREN FATHER--MOTHER
REGULAR, COMMERCE            NO COVID CONTACT   Negative   Negative
## 16           WITH CHILDREN         FATHER
REGULAR, COMMERCE            NO COVID CONTACT   Positive   Negative
## 17           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE            NO COVID CONTACT   Negative   Positive
## 18           WITH CHILDREN         FATHER
REGULAR, COMMERCE            NO COVID CONTACT   Positive   Negative
## 19           WITH CHILDREN         FATHER
REGULAR, COMMERCE            NO COVID CONTACT   Negative   Negative
## 20           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE            NO COVID CONTACT   Positive   Negative
## 21           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE            NO COVID CONTACT   Negative   Negative
## 22           WITH CHILDREN         FATHER
REGULAR, FORMAL           NO COVID CONTACT   Negative   Negative
## 23           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE            NO COVID CONTACT   Positive   Negative
## 24           WITH CHILDREN         FATHER
```

```
REGULAR, COMMERCE UNSURE ABOUT COVID CONTACT   Negative   Negative
## 25           WITH CHILDREN    NO RESPONSE
REGULAR, COMMERCE           NO COVID CONTACT   Positive   Negative
## 26           WITH CHILDREN         FATHER
REGULAR, FORMAL           NO COVID CONTACT   Negative   Positive
## 27           WITH CHILDREN         FATHER
REGULAR, FORMAL           NO COVID CONTACT   Positive   Negative
## 28           WITH CHILDREN         FATHER
REGULAR, FORMAL           NO COVID CONTACT   Negative   Positive
##                                                 symptoms    consultation
treatment_combinations               drugsource
## 1                                             Muscle pain            <NA>
Paracetamol         Self or familial
## 2                                             No symptoms            <NA>
<NA>                    <NA>
## 3                                             No symptoms            <NA>
<NA>                    <NA>
## 4                 Rhinitis--Sneezing--Anosmia or ageusia             <NA>
Antibiotics         Self or familial
## 5                                             No symptoms            <NA>
<NA>                    <NA>
## 6  Fever--Cough--Rhinitis--Nausea or vomiting--Diarrhoea  Private clinic
Paracetamol--Antibiotics               Pharmacist
## 7                                             Sneezing               <NA>
Traditional meds.         Self or familial
## 8                                             Headache               <NA>
Paracetamol         Self or familial
## 9                           Rhinitis--Sneezing--Headache             <NA>
Paracetamol         Self or familial
## 10                    Fever--Rhinitis--Anosmia or ageusia            <NA>
Paracetamol--Traditional meds. Doctor--Self or familial
## 11                                            No symptoms            <NA>
<NA>                    <NA>
## 12                                            No symptoms            <NA>
<NA>                    <NA>
## 13                                       Cough--Headache             <NA>
Paracetamol--Antibiotics         Self or familial
## 14                                            No symptoms            <NA>
<NA>                    <NA>
## 15                                            No symptoms            <NA>
<NA>                    <NA>
## 16                                            No symptoms            <NA>
<NA>                    <NA>
## 17                      Fever--Cough--Rhinitis--Headache Public hospital
Paracetamol--Antibiotics--Other                    Doctor
## 18                                            No symptoms            <NA>
<NA>                    <NA>
## 19                                            No symptoms            <NA>
<NA>                    <NA>
## 20                                            No symptoms            <NA>
<NA>                    <NA>
## 21                                       Cough--Rhinitis             <NA>
Traditional meds.         Self or familial
## 22                                                 Fever  Private clinic
Paracetamol--Hydroxychloroquine               Pharmacist
## 23                                       Cough--Headache Public hospital
```

```
Paracetamol--Antibiotics                    Doctor
## 24                              Cough--Rhinitis--Headache             <NA>
No medication                      <NA>
## 25                                      Fever--Diarrhoea             <NA>
No medication                      <NA>
## 26                              Cough--Rhinitis--Sneezing             <NA>
No medication                      <NA>
## 27                                       Fever--Headache             <NA>
Paracetamol         Self or familial
## 28                                           No symptoms             <NA>
<NA>                      <NA>
##                 hospitalised sequelae respiration_frequency drug_parac
drug_antibio drug_hydrocortisone drug_other_anti_inflam drug_antiviral
## 1                          No     <NA>                     NA           1
0                     0                         0               0
## 2                          No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 3  Yes, but not COVID-linked     <NA>                       NA         NA
NA                    NA                        NA              NA
## 4                          No     <NA>                       NA          0
1                     0                         0               0
## 5                          No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 6                          No     <NA>                       NA          1
1                     0                         0               0
## 7                          No     <NA>                       NA          0
0                     0                         0               0
## 8                          No     <NA>                       NA          1
0                     0                         0               0
## 9                          No     <NA>                       NA          1
0                     0                         0               0
## 10                         No     <NA>                       NA          1
0                     0                         0               0
## 11                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 12                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 13                         No     <NA>                       NA          1
1                     0                         0               0
## 14                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 15                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 16                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 17                         No     <NA>                       NA          1
1                     0                         0               0
## 18                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 19                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 20                         No     <NA>                       NA         NA
NA                    NA                        NA              NA
## 21                         No     <NA>                       NA          0
0                     0                         0               0
## 22 Yes, but not COVID-linked     <NA>                       NA          1
```

```
0                    0                         0               0
## 23                          No      <NA>                    NA           1
1                    0                         0               0
## 24                          No      <NA>                    NA           0
0                    0                         0               0
## 25                          No      <NA>                    NA           0
0                    0                         0               0
## 26                          No      <NA>                    NA           0
0                    0                         0               0
## 27                          No      <NA>                    NA           1
0                    0                         0               0
## 28                 No response      <NA>                    NA          NA
NA                   NA                        NA              NA
##     drug_chloro drug_tradn drug_oxygen drug_other drug_no_resp drug_none
## 1             0          0           0          0            0         0
## 2            NA         NA          NA         NA           NA        NA
## 3            NA         NA          NA         NA           NA        NA
## 4             0          0           0          0            0         0
## 5            NA         NA          NA         NA           NA        NA
## 6             0          0           0          0            0         0
## 7             0          1           0          0            0         0
## 8             0          0           0          0            0         0
## 9             0          0           0          0            0         0
## 10            0          1           0          0            0         0
## 11           NA         NA          NA         NA           NA        NA
## 12           NA         NA          NA         NA           NA        NA
## 13            0          0           0          0            0         0
## 14           NA         NA          NA         NA           NA        NA
## 15           NA         NA          NA         NA           NA        NA
## 16           NA         NA          NA         NA           NA        NA
## 17            0          0           0          1            0         0
## 18           NA         NA          NA         NA           NA        NA
## 19           NA         NA          NA         NA           NA        NA
## 20           NA         NA          NA         NA           NA        NA
## 21            0          1           0          0            0         0
## 22            1          0           0          0            0         0
## 23            0          0           0          0            0         0
## 24            0          0           0          0            0         1
## 25            0          0           0          0            0         1
## 26            0          0           0          0            0         1
## 27            0          0           0          0            0         0
## 28           NA         NA          NA         NA           NA        NA
##  [ reached 'max' / getOption("max.print") -- omitted 943 rows ]
```

Use {dplyr's} `case_when()` if you are re-coding into many new groups, or if you need to use complex logic statements to re-code values.

This function evaluates every row in the data frame, assess whether the rows meets specified criteria, and assigns the correct new value.

3. The individual with `unique_key = 75` is described as "Housewhold head" in the variable `positioninthehousehold`. Replace this with "Household head"

String inconsistencies

gsub

When fixing inconsistencies and structural issues in variables, it is often necessary to selectively replace multiple occurrences of a text within an R string.

The function `gsub()` is ideal for this as it searches for a regular expression in a string and replaces it.

The general syntax of the function is:

```
gsub(search_term, replacement_term, string_searched)
```

When combined with {dplyr's} `mutate()` function, a column of a data frame can be cleaned to enable analysis.

We will illustrate the use of this function using the `education` variable.

```
yaounde %>% count(education, name = "Count")
```

```
##                education Count
## 1             Doctorate1    17
## 2 No formal instruction1    52
## 3           No response1     4
## 4                 Other1     2
## 5               Primary1   318
## 6             Secondary1   433
## 7            University1   145
```

There is an unwanted element in each of the categories ("1"). We will remove this element using `gsub` and replace it with an empty string "".

```
yaounde %>%
    select(education)%>%
    mutate(education_new = gsub("1","",education))
```

```
##                 education       education_new
## 1             Secondary1           Secondary
## 2            University1          University
## 3            University1          University
```

```
## 4                Secondary1              Secondary
## 5                  Primary1                Primary
## 6                Secondary1              Secondary
## 7                Secondary1              Secondary
## 8                Doctorate1              Doctorate
## 9                Secondary1              Secondary
## 10               Secondary1              Secondary
## 11               Secondary1              Secondary
## 12                 Primary1                Primary
## 13                 Primary1                Primary
## 14                 Primary1                Primary
## 15               Secondary1              Secondary
## 16                 Primary1                Primary
## 17                 Primary1                Primary
## 18                 Primary1                Primary
## 19                 Primary1                Primary
## 20                 Primary1                Primary
## 21                 Primary1                Primary
## 22               Secondary1              Secondary
## 23                 Primary1                Primary
## 24              University1             University
## 25                 Primary1                Primary
## 26              University1             University
## 27                 Primary1                Primary
## 28               Secondary1              Secondary
## 29               Secondary1              Secondary
## 30               Secondary1              Secondary
## 31                 Primary1                Primary
## 32               Secondary1              Secondary
## 33              University1             University
## 34               Secondary1              Secondary
## 35                 Primary1                Primary
## 36               Secondary1              Secondary
## 37              University1             University
## 38               Secondary1              Secondary
## 39               Secondary1              Secondary
## 40               Secondary1              Secondary
## 41               Secondary1              Secondary
## 42               Secondary1              Secondary
## 43                 Primary1                Primary
## 44                 Primary1                Primary
## 45               Secondary1              Secondary
## 46               Secondary1              Secondary
## 47               Secondary1              Secondary
## 48               Secondary1              Secondary
## 49                 Primary1                Primary
## 50                 Primary1                Primary
## 51                 Primary1                Primary
## 52                 Primary1                Primary
## 53  No formal instruction1 No formal instruction
## 54               Secondary1              Secondary
## 55                 Primary1                Primary
## 56                 Primary1                Primary
## 57               Doctorate1              Doctorate
## 58              University1             University
## 59                 Primary1                Primary
```

```
## 60              Secondary1              Secondary
## 61              Secondary1              Secondary
## 62            University1            University
## 63              Secondary1              Secondary
## 64              Secondary1              Secondary
## 65                Primary1                Primary
## 66              Secondary1              Secondary
## 67              Secondary1              Secondary
## 68                Primary1                Primary
## 69              Secondary1              Secondary
## 70              Secondary1              Secondary
## 71                Primary1                Primary
## 72                Primary1                Primary
## 73            University1            University
## 74                Primary1                Primary
## 75  No formal instruction1 No formal instruction
## 76  No formal instruction1 No formal instruction
## 77  No formal instruction1 No formal instruction
## 78  No formal instruction1 No formal instruction
## 79              Secondary1              Secondary
## 80              Secondary1              Secondary
## 81              Secondary1              Secondary
## 82  No formal instruction1 No formal instruction
## 83              Secondary1              Secondary
## 84              Secondary1              Secondary
## 85                Primary1                Primary
## 86              Secondary1              Secondary
## 87              Secondary1              Secondary
## 88              Secondary1              Secondary
## 89              Secondary1              Secondary
## 90              Secondary1              Secondary
## 91              Secondary1              Secondary
## 92              Secondary1              Secondary
## 93              Secondary1              Secondary
## 94                Primary1                Primary
## 95              Secondary1              Secondary
## 96              Secondary1              Secondary
## 97                Primary1                Primary
## 98                Primary1                Primary
## 99              Secondary1              Secondary
## 100               Primary1                Primary
## 101               Primary1                Primary
## 102           University1            University
## 103               Primary1                Primary
## 104             Secondary1              Secondary
## 105             Secondary1              Secondary
## 106             Secondary1              Secondary
## 107             Secondary1              Secondary
## 108               Primary1                Primary
## 109               Primary1                Primary
## 110             Secondary1              Secondary
## 111             Secondary1              Secondary
## 112               Primary1                Primary
## 113 No formal instruction1 No formal instruction
## 114               Primary1                Primary
## 115               Primary1                Primary
```

```
## 116              Primary1              Primary
## 117              Primary1              Primary
## 118              Primary1              Primary
## 119              Primary1              Primary
## 120              Primary1              Primary
## 121            Secondary1            Secondary
## 122              Primary1              Primary
## 123            Secondary1            Secondary
## 124              Primary1              Primary
## 125              Primary1              Primary
## 126            University1           University
## 127              Primary1              Primary
## 128            Secondary1            Secondary
## 129            Secondary1            Secondary
## 130              Primary1              Primary
## 131            University1           University
## 132            University1           University
## 133            Secondary1            Secondary
## 134            University1           University
## 135            University1           University
## 136            University1           University
## 137            Secondary1            Secondary
## 138            Secondary1            Secondary
## 139            Secondary1            Secondary
## 140            University1           University
## 141              Primary1              Primary
## 142              Primary1              Primary
## 143              Primary1              Primary
## 144            Secondary1            Secondary
## 145              Primary1              Primary
## 146            Secondary1            Secondary
## 147            Secondary1            Secondary
## 148              Primary1              Primary
## 149 No formal instruction1 No formal instruction
## 150 No formal instruction1 No formal instruction
## 151            Secondary1            Secondary
## 152            University1           University
## 153            University1           University
## 154            Secondary1            Secondary
## 155            Secondary1            Secondary
## 156            Secondary1            Secondary
## 157            Secondary1            Secondary
## 158              Primary1              Primary
## 159              Primary1              Primary
## 160            Secondary1            Secondary
## 161            Secondary1            Secondary
## 162            Secondary1            Secondary
## 163            University1           University
## 164            University1           University
## 165            Secondary1            Secondary
## 166            Secondary1            Secondary
## 167 No formal instruction1 No formal instruction
## 168              Primary1              Primary
## 169            Secondary1            Secondary
## 170            Secondary1            Secondary
## 171            Secondary1            Secondary
```

```
## 172          Primary1          Primary
## 173        Secondary1        Secondary
## 174          Primary1          Primary
## 175       University1       University
## 176       University1       University
## 177        Secondary1        Secondary
## 178        Secondary1        Secondary
## 179        Secondary1        Secondary
## 180       University1       University
## 181        Secondary1        Secondary
## 182       University1       University
## 183        Secondary1        Secondary
## 184        Secondary1        Secondary
## 185        Secondary1        Secondary
## 186          Primary1          Primary
## 187        Secondary1        Secondary
## 188        Secondary1        Secondary
## 189       University1       University
## 190        Secondary1        Secondary
## 191          Primary1          Primary
## 192        Secondary1        Secondary
## 193          Primary1          Primary
## 194        Secondary1        Secondary
## 195        Secondary1        Secondary
## 196        Secondary1        Secondary
## 197        Secondary1        Secondary
## 198        Secondary1        Secondary
## 199          Primary1          Primary
## 200       University1       University
## 201        Secondary1        Secondary
## 202       University1       University
## 203       University1       University
## 204        Secondary1        Secondary
## 205          Primary1          Primary
## 206        Secondary1        Secondary
## 207        Secondary1        Secondary
## 208        Secondary1        Secondary
## 209        Secondary1        Secondary
## 210          Primary1          Primary
## 211        Secondary1        Secondary
## 212          Primary1          Primary
## 213        Secondary1        Secondary
## 214       University1       University
## 215        Secondary1        Secondary
## 216          Primary1          Primary
## 217          Primary1          Primary
## 218        Secondary1        Secondary
## 219        Secondary1        Secondary
## 220          Primary1          Primary
## 221        Secondary1        Secondary
## 222        Secondary1        Secondary
## 223        Secondary1        Secondary
## 224       University1       University
## 225        Secondary1        Secondary
## 226        Secondary1        Secondary
## 227        Secondary1        Secondary
```

```
## 228             Secondary1             Secondary
## 229             University1            University
## 230             Secondary1             Secondary
## 231               Primary1               Primary
## 232               Primary1               Primary
## 233               Primary1               Primary
## 234 No formal instruction1 No formal instruction
## 235             University1            University
## 236             Secondary1             Secondary
## 237             Secondary1             Secondary
## 238             Secondary1             Secondary
## 239             Secondary1             Secondary
## 240             Secondary1             Secondary
## 241           No response1           No response
## 242               Primary1               Primary
## 243             University1            University
## 244               Primary1               Primary
## 245             Secondary1             Secondary
## 246             Secondary1             Secondary
## 247             Secondary1             Secondary
## 248               Primary1               Primary
## 249             Secondary1             Secondary
## 250             Secondary1             Secondary
## 251             Secondary1             Secondary
## 252               Primary1               Primary
## 253             Secondary1             Secondary
## 254               Primary1               Primary
## 255           No response1           No response
## 256               Primary1               Primary
## 257               Primary1               Primary
## 258             Secondary1             Secondary
## 259             Secondary1             Secondary
## 260             University1            University
## 261               Primary1               Primary
## 262             Secondary1             Secondary
## 263             Secondary1             Secondary
## 264               Primary1               Primary
## 265               Primary1               Primary
## 266             Secondary1             Secondary
## 267             Secondary1             Secondary
## 268             Secondary1             Secondary
## 269             University1            University
## 270             Secondary1             Secondary
## 271             Secondary1             Secondary
## 272               Primary1               Primary
## 273             Secondary1             Secondary
## 274             Secondary1             Secondary
## 275             University1            University
## 276               Primary1               Primary
## 277             Secondary1             Secondary
## 278             Secondary1             Secondary
## 279               Primary1               Primary
## 280 No formal instruction1 No formal instruction
## 281             Secondary1             Secondary
## 282               Primary1               Primary
## 283               Primary1               Primary
```

```
## 284          Secondary1          Secondary
## 285            Primary1            Primary
## 286            Primary1            Primary
## 287            Primary1            Primary
## 288          Secondary1          Secondary
## 289          Secondary1          Secondary
## 290         University1         University
## 291          Secondary1          Secondary
## 292          Secondary1          Secondary
## 293            Primary1            Primary
## 294            Primary1            Primary
## 295            Primary1            Primary
## 296            Primary1            Primary
## 297       No response1       No response
## 298            Primary1            Primary
## 299          Secondary1          Secondary
## 300            Primary1            Primary
## 301          Secondary1          Secondary
## 302          Secondary1          Secondary
## 303          Secondary1          Secondary
## 304          Secondary1          Secondary
## 305         University1         University
## 306          Secondary1          Secondary
## 307            Primary1            Primary
## 308          Secondary1          Secondary
## 309          Secondary1          Secondary
## 310          Secondary1          Secondary
## 311            Primary1            Primary
## 312            Primary1            Primary
## 313          Secondary1          Secondary
## 314          Secondary1          Secondary
## 315            Primary1            Primary
## 316            Primary1            Primary
## 317            Primary1            Primary
## 318            Primary1            Primary
## 319            Primary1            Primary
## 320          Secondary1          Secondary
## 321          Secondary1          Secondary
## 322          Doctorate1          Doctorate
## 323          Secondary1          Secondary
## 324         University1         University
## 325         University1         University
## 326          Secondary1          Secondary
## 327          Secondary1          Secondary
## 328         University1         University
## 329          Secondary1          Secondary
## 330          Secondary1          Secondary
## 331            Primary1            Primary
## 332          Secondary1          Secondary
## 333            Primary1            Primary
## 334            Primary1            Primary
## 335          Secondary1          Secondary
## 336         University1         University
## 337          Secondary1          Secondary
## 338          Secondary1          Secondary
## 339          Secondary1          Secondary
```

```
## 340          Secondary1          Secondary
## 341            Primary1            Primary
## 342            Primary1            Primary
## 343          Secondary1          Secondary
## 344          Secondary1          Secondary
## 345          Secondary1          Secondary
## 346            Primary1            Primary
## 347          Secondary1          Secondary
## 348            Primary1            Primary
## 349          Secondary1          Secondary
## 350         University1         University
## 351          Secondary1          Secondary
## 352          Secondary1          Secondary
## 353          Secondary1          Secondary
## 354          Secondary1          Secondary
## 355          Secondary1          Secondary
## 356          Secondary1          Secondary
## 357         University1         University
## 358         University1         University
## 359            Primary1            Primary
## 360          Secondary1          Secondary
## 361          Secondary1          Secondary
## 362            Primary1            Primary
## 363          Secondary1          Secondary
## 364          Secondary1          Secondary
## 365            Primary1            Primary
## 366          Secondary1          Secondary
## 367          Secondary1          Secondary
## 368          Secondary1          Secondary
## 369          Secondary1          Secondary
## 370          Secondary1          Secondary
## 371         University1         University
## 372          Secondary1          Secondary
## 373          Secondary1          Secondary
## 374          Secondary1          Secondary
## 375            Primary1            Primary
## 376            Primary1            Primary
## 377            Primary1            Primary
## 378          Secondary1          Secondary
## 379          Secondary1          Secondary
## 380          Doctorate1          Doctorate
## 381         University1         University
## 382         University1         University
## 383          Secondary1          Secondary
## 384          Secondary1          Secondary
## 385          Secondary1          Secondary
## 386          Secondary1          Secondary
## 387         University1         University
## 388         University1         University
## 389            Primary1            Primary
## 390          Secondary1          Secondary
## 391         University1         University
## 392          Secondary1          Secondary
## 393         University1         University
## 394          Secondary1          Secondary
## 395         University1         University
```

```
## 396              Secondary1              Secondary
## 397              Secondary1              Secondary
## 398              Secondary1              Secondary
## 399             University1             University
## 400              Secondary1              Secondary
## 401                Primary1                Primary
## 402              Secondary1              Secondary
## 403              Secondary1              Secondary
## 404             University1             University
## 405                Primary1                Primary
## 406              Secondary1              Secondary
## 407                Primary1                Primary
## 408              Secondary1              Secondary
## 409                Primary1                Primary
## 410                Primary1                Primary
## 411                Primary1                Primary
## 412              Secondary1              Secondary
## 413              Secondary1              Secondary
## 414              Secondary1              Secondary
## 415                Primary1                Primary
## 416                Primary1                Primary
## 417                Primary1                Primary
## 418                Primary1                Primary
## 419                Primary1                Primary
## 420              Secondary1              Secondary
## 421              Secondary1              Secondary
## 422              Secondary1              Secondary
## 423              Secondary1              Secondary
## 424                Primary1                Primary
## 425              Secondary1              Secondary
## 426                Primary1                Primary
## 427                Primary1                Primary
## 428              Secondary1              Secondary
## 429              Secondary1              Secondary
## 430                Primary1                Primary
## 431                Primary1                Primary
## 432              Secondary1              Secondary
## 433                Primary1                Primary
## 434              Secondary1              Secondary
## 435              Secondary1              Secondary
## 436              Secondary1              Secondary
## 437              Secondary1              Secondary
## 438                Primary1                Primary
## 439                Primary1                Primary
## 440 No formal instruction1 No formal instruction
## 441             University1             University
## 442                Primary1                Primary
## 443                Primary1                Primary
## 444                Primary1                Primary
## 445                Primary1                Primary
## 446                Primary1                Primary
## 447                Primary1                Primary
## 448                Primary1                Primary
## 449                Primary1                Primary
## 450                Primary1                Primary
## 451              Secondary1              Secondary
```

```
## 452                Primary1                Primary
## 453 No formal instruction1 No formal instruction
## 454              Secondary1              Secondary
## 455              Secondary1              Secondary
## 456                Primary1                Primary
## 457                Primary1                Primary
## 458                Primary1                Primary
## 459              Secondary1              Secondary
## 460              Secondary1              Secondary
## 461             University1             University
## 462                Primary1                Primary
## 463                Primary1                Primary
## 464              Secondary1              Secondary
## 465              Secondary1              Secondary
## 466                Primary1                Primary
## 467              Secondary1              Secondary
## 468              Secondary1              Secondary
## 469              Secondary1              Secondary
## 470             University1             University
## 471                Primary1                Primary
## 472             University1             University
## 473              Secondary1              Secondary
## 474             University1             University
## 475              Secondary1              Secondary
## 476                Primary1                Primary
## 477 No formal instruction1 No formal instruction
## 478 No formal instruction1 No formal instruction
## 479              Secondary1              Secondary
## 480                Primary1                Primary
## 481              Secondary1              Secondary
## 482                Primary1                Primary
## 483                Primary1                Primary
## 484                Primary1                Primary
## 485              Secondary1              Secondary
## 486              Secondary1              Secondary
## 487                Primary1                Primary
## 488                Primary1                Primary
## 489                Primary1                Primary
## 490                Primary1                Primary
## 491              Secondary1              Secondary
## 492 No formal instruction1 No formal instruction
## 493                Primary1                Primary
## 494              Secondary1              Secondary
## 495                Primary1                Primary
## 496                Primary1                Primary
## 497              Secondary1              Secondary
## 498              Secondary1              Secondary
## 499                Primary1                Primary
## 500              Doctorate1              Doctorate
##  [ reached 'max' / getOption("max.print") -- omitted 471 rows ]
```

Let us also change the encoding of `treatment_combinations` and `occupation` and replace all `--` with `/`

```
yaounde %>%
    select(treatment_combinations,occupation)%>%
    mutate(across(c(treatment_combinations,occupation),
           ~ gsub("--","/",.x)))
```

```
##
treatment_combinations                occupation
## 1
Paracetamol            Informal worker
## 2
<NA>            Salaried worker
## 3
<NA>                    Student
## 4
Antibiotics                    Student
## 5
<NA>              Trader/Farmer
## 6
Paracetamol/Antibiotics                    Student
## 7                                                          Traditional
meds.                    Student
## 8
Paracetamol                    Student
## 9
Paracetamol                    Student
## 10                                          Paracetamol/Traditional
meds.                     Trader
## 11
<NA>                      Other
## 12
<NA>                 Unemployed
## 13
Paracetamol/Antibiotics                     Trader
## 14
<NA>                 Home-maker
## 15
<NA>                     Trader
## 16
<NA>            Informal worker
## 17
Paracetamol/Antibiotics/Other            Informal worker
## 18
<NA>                    Student
## 19
<NA>                    Student
## 20
<NA>                    Student
## 21                                                         Traditional
meds.                     Student
## 22
Paracetamol/Hydroxychloroquine                    Student
## 23
Paracetamol/Antibiotics                     Trader
## 24                                                             No
```

```
medication             Informal worker
## 25                                                                  No
medication                     Student
## 26                                                                  No
medication             Salaried worker
## 27
Paracetamol                  Home-maker
## 28
<NA>                       Student
## 29
<NA>                         Other
## 30                                          Paracetamol/Traditional
meds./Other                        Trader
## 31
<NA>                       Student
## 32
<NA>                       Student
## 33                                          Antibiotics/Traditional
meds./Other                        Retired
## 34
<NA>                       Retired
## 35
<NA>                        Trader
## 36
Paracetamol/Other                        Trader
## 37
<NA>                    Unemployed
## 38
<NA>              Informal worker
## 39                                                        Traditional
meds.                    Home-maker
## 40
<NA>              Informal worker
## 41
<NA>                    Home-maker
## 42
<NA>              Informal worker
## 43
<NA>                       Student
## 44
<NA>                       Student
## 45
<NA>                       Student
## 46
<NA>    Retired/Informal worker
## 47                                              Paracetamol/Traditional
meds.                       Retired
## 48
<NA> Home-maker/Informal worker
## 49
<NA>                       Trader
## 50
<NA>            Trader/Unemployed
## 51
<NA>                       Trader
## 52
```

```
<NA>             Informal worker
## 53
<NA>             Informal worker
## 54
<NA>                  Unemployed
## 55                                                          Traditional
meds.                  Home-maker
## 56
<NA>                  Home-maker
## 57
<NA>                  Unemployed
## 58
<NA>             Informal worker
## 59
<NA>                  Home-maker
## 60
<NA>                     Student
## 61                                  Paracetamol/Antibiotics/Traditional
meds.             Informal worker
## 62
Paracetamol/Antibiotics                 Unemployed
## 63
<NA>                  Home-maker
## 64
<NA>                  Home-maker
## 65
<NA>                     Retired
## 66
<NA>                      Trader
## 67
<NA>                     Student
## 68
<NA>                     Student
## 69
<NA>             Informal worker
## 70
<NA>             Informal worker
## 71
Paracetamol/Antibiotics                    Student
## 72
<NA>                     Student
## 73
<NA>                  Unemployed
## 74
<NA>                  Unemployed
## 75
<NA>                      Trader
## 76
<NA>     Informal worker/Trader
## 77
<NA>             Informal worker
## 78
<NA>             Informal worker
## 79
<NA>             Informal worker
## 80
```

```
<NA>                  Home-maker
## 81
<NA>                     Student
## 82
<NA>                  Home-maker
## 83
<NA>                  Unemployed
## 84
<NA>                     Student
## 85
<NA>                       Other
## 86                                    Paracetamol/Antibiotics/Traditional
meds.             Salaried worker
## 87
<NA>                       Other
## 88
<NA>             Informal worker
## 89
<NA>             Informal worker
## 90
<NA>                  Unemployed
## 91
<NA>             Salaried worker
## 92
<NA>                  Unemployed
## 93                                                                    No
medication              Informal worker
## 94
<NA>                  Unemployed
## 95
<NA>                     Student
## 96
<NA>             Informal worker
## 97
Paracetamol             Informal worker
## 98
<NA>                  Home-maker
## 99
<NA>                       Trader
## 100
<NA>             Informal worker
## 101
<NA>             Informal worker
## 102
<NA>             Informal worker
## 103
<NA>                       Trader
## 104
<NA>                       Trader
## 105
<NA>             Informal worker
## 106
<NA>                     Student
## 107
<NA>             Salaried worker
## 108
```

```
<NA>                     Student
## 109
<NA>             Informal worker
## 110
<NA>                  Home-maker
## 111
<NA>                     Student
## 112
<NA>                     Student
## 113
<NA>                      Trader
## 114
<NA>             Informal worker
## 115
<NA>             Informal worker
## 116
<NA>                      Trader
## 117
<NA>                     Student
## 118
<NA>                      Trader
## 119
Paracetamol                    Student
## 120
<NA>             Informal worker
## 121
Paracetamol/Other                    Trader
## 122
<NA>                     Student
## 123
<NA>                     Retired
## 124
<NA>                     Student
## 125
<NA>                     Student
## 126
<NA>                     Student
## 127
<NA>    Student/Informal worker
## 128
<NA>    Student/Informal worker
## 129
<NA>                  Unemployed
## 130
Paracetamol/Other    Student/Informal worker
## 131
<NA>             Salaried worker
## 132
Paracetamol                    Student
## 133
<NA>                     Retired
## 134                                                                No
medication                   Student
## 135
<NA>             Informal worker
## 136
```

```
Other                    Student
## 137
<NA>             Informal worker
## 138
<NA>                     Student
## 139
<NA>                     Student
## 140
Antibiotics                 Unemployed
## 141
<NA>                      Trader
## 142
Other                      Trader
## 143                                                                  No
response                    Trader
## 144
<NA>                     Student
## 145                                                          Traditional
meds.                  Home-maker
## 146
<NA>                     Student
## 147
<NA>                     Student
## 148
<NA>                  Home-maker
## 149
Other                  Home-maker
## 150
<NA>                       Other
## 151
<NA>             Salaried worker
## 152
<NA>             Salaried worker
## 153                                                              No
medication             Informal worker
## 154
<NA>                      Trader
## 155
<NA>                     Student
## 156
<NA>             Informal worker
## 157
<NA>             Informal worker
## 158
<NA>                  Home-maker
## 159
Other                    Student
## 160
<NA>                     Student
## 161
<NA>             Informal worker
## 162
<NA>             Informal worker
## 163
<NA>             Informal worker
## 164
```

```
<NA>                Home-maker
## 165
<NA>                   Student
## 166
<NA>                Unemployed
## 167
<NA>               No response
## 168
<NA>            Informal worker
## 169
<NA>                    Farmer
## 170
<NA>                   Student
## 171                                                    Traditional
meds.              Student/Other
## 172                                                    Traditional
meds.                   Student
## 173                                                    Traditional
meds.                Unemployed
## 174
<NA>                    Trader
## 175
<NA>                   Student
## 176
<NA>                   Student
## 177
<NA>                   Student
## 178
<NA>                Home-maker
## 179                                                    Traditional
meds.                    Farmer
## 180                                                    Traditional
meds.                   Student
## 181
<NA>            Informal worker
## 182
<NA>                 Home-maker
## 183
<NA>            Informal worker
## 184                                                         No
medication           Informal worker
## 185
<NA>                 Home-maker
## 186                                          Paracetamol/Traditional
meds.            Informal worker
## 187
<NA>                    Other
## 188
<NA>                   Student
## 189
<NA>                   Student
## 190
Paracetamol          Informal worker
## 191                                                       No
response                 Student
## 192                                                       No
```

```
response                      Trader
## 193
<NA>                  Home-maker
## 194
<NA>                  Home-maker
## 195
<NA>                     Student
## 196
Paracetamol                       Trader
## 197
<NA>                  Home-maker
## 198
<NA>                      Trader
## 199
<NA>                     Student
## 200
<NA>             Salaried worker
## 201
<NA>                      Trader
## 202
<NA>                  Unemployed
## 203
<NA>                  Unemployed
## 204
<NA>             Informal worker
## 205
<NA>                       Other
## 206
<NA>                  Home-maker
## 207
<NA>                     Student
## 208
<NA>                     Student
## 209
<NA>             Informal worker
## 210                                                              Traditional
meds.                     Student
## 211
<NA>                     Student
## 212
<NA>                  Home-maker
## 213                                  Paracetamol/Antibiotics/Traditional
meds.       Informal worker/Other
## 214
<NA>     Student/Informal worker
## 215
<NA>                  Home-maker
## 216
<NA>                      Trader
## 217
Other              Informal worker
## 218                Paracetamol/Antibiotics/Other anti-inflamm./Traditional
meds.                Farmer/Other
## 219
<NA>                       Other
## 220
```

```
<NA>            Informal worker
## 221
<NA>            Informal worker
## 222                                    Paracetamol/Antibiotics/Traditional
meds.    Student/Informal worker
## 223
<NA>                    Trader
## 224
<NA>                   Student
## 225
<NA>                     Other
## 226
<NA>            Informal worker
## 227
<NA>                    Trader
## 228
<NA>                    Trader
## 229
Antibiotics            Informal worker
## 230
<NA>                   Student
## 231
<NA>                   Student
## 232
<NA>                   Student
## 233
<NA>                   Student
## 234
<NA>                Home-maker
## 235
<NA>                   Student
## 236                                                        Traditional
meds./Other            Salaried worker
## 237
<NA>                Home-maker
## 238                                                        Traditional
meds./Other                Unemployed
## 239
<NA>                    Trader
## 240
<NA>                   Student
## 241
<NA>               No response
## 242
<NA>                   Student
## 243
<NA>            Informal worker
## 244
<NA>            Informal worker
## 245
<NA>     Informal worker/Trader
## 246
<NA>            Informal worker
## 247
<NA>            Informal worker
## 248
```

```
<NA>                  Home-maker
## 249
<NA>             Informal worker
## 250
<NA>                     Trader
## 251
<NA>                  Unemployed
## 252
<NA>             Informal worker
## 253
<NA>                     Student
## 254
<NA>                  Home-maker
## 255
<NA>                     Student
## 256
<NA>                     Student
## 257
<NA>                 No response
## 258
<NA>                     Student
## 259
Paracetamol/Antibiotics            Informal worker
## 260                                                Paracetamol/Traditional
meds.             Informal worker
## 261                              Paracetamol/Antibiotics/Traditional
meds./Oxygen             Informal worker
## 262
<NA>                     Student
## 263
<NA>                     Student
## 264
<NA>                     Student
## 265
<NA>                     Student
## 266
<NA>             Informal worker
## 267
<NA>                      Trader
## 268
<NA>                      Trader
## 269
<NA>             Informal worker
## 270
<NA>             Informal worker
## 271
<NA>                     Student
## 272
<NA>                  Home-maker
## 273
<NA>                     Student
## 274
Paracetamol                  Home-maker
## 275
Paracetamol/Antibiotics/Other                  Unemployed
## 276
```

```
<NA>                      Trader
## 277
<NA>              Informal worker
## 278
<NA>                      Student
## 279
<NA>                      Student
## 280
<NA>                  No response
## 281
<NA>                      Student
## 282
<NA>                      Student
## 283
<NA>              Informal worker
## 284
<NA>                      Student
## 285
<NA>                       Trader
## 286
<NA>                      Student
## 287
<NA>                      Student
## 288
<NA>                      Student
## 289
<NA>              Informal worker
## 290
<NA>                      Student
## 291
<NA>                    Unemployed
## 292
<NA>              Informal worker
## 293
Antibiotics                  Home-maker
## 294
<NA>                      Student
## 295                                  Paracetamol/Antibiotics/Other anti-
inflamm.                     Student
## 296
Paracetamol                     Student
## 297                                                                  No
medication             Informal worker
## 298
<NA>                      Student
## 299
<NA>                      Student
## 300
<NA>                      Student
## 301
<NA>                      Student
## 302 Paracetamol/Antibiotics/Hydrocortisone/Other anti-inflamm./Traditional
meds.                      Student
## 303                                                        Traditional
meds.                       Trader
## 304
```

```
Paracetamol/Hydrocortisone          Home-maker/Trader
## 305                                                                  Traditional
meds.            Salaried worker
## 306
<NA>                    Student
## 307                                                                  Traditional
meds.                     Trader
## 308                                                                      No
medication            Salaried worker
## 309
<NA>                 Unemployed
## 310
<NA>                    Student
## 311
<NA>                    Student
## 312
<NA>                 Unemployed
## 313
<NA>                    Student
## 314
<NA>                    Student
## 315
<NA>                    Student
## 316
Other                    Student
## 317
<NA>                     Trader
## 318
<NA>                    Student
## 319
Paracetamol/Other                    Student
## 320
<NA>                    Student
## 321
<NA>            Informal worker
## 322
<NA>            Salaried worker
## 323
<NA>            Informal worker
## 324
<NA>            Informal worker
## 325
<NA>                    Student
## 326
<NA>                    Student
## 327
Paracetamol/Other                    Student
## 328
<NA>            Informal worker
## 329                           Paracetamol/Antibiotics/Other anti-
inflamm./Other                   Retired
## 330
<NA>                    Retired
## 331
<NA>                 Home-maker
## 332
```

```
<NA>            Informal worker
## 333
<NA>                    Trader
## 334
<NA>                Unemployed
## 335
<NA>            Informal worker
## 336
<NA>                    Student
## 337
<NA>            Informal worker
## 338
<NA>                Unemployed
## 339
<NA>                    Student
## 340
<NA>                    Student
## 341
<NA>                    Trader
## 342
Other            Informal worker
## 343
<NA>                    Retired
## 344                                                    Traditional
meds.            Salaried worker
## 345
<NA>                    Trader
## 346                                                Other anti-
inflamm.                   Retired
## 347                                                         No
medication                    Trader
## 348
<NA>               No response
## 349                                                    Traditional
meds.                    Trader
## 350
Paracetamol/Antibiotics                    Trader
## 351
<NA>            Informal worker
## 352
<NA>            Informal worker
## 353
<NA>                    Trader
## 354
Antibiotics           Informal worker
## 355                                                         No
medication                   Student
## 356
<NA>                    Student
## 357                                 Paracetamol/Antibiotics/Traditional
meds.            Salaried worker
## 358
<NA>                    Student
## 359                                                         No
medication                    Trader
## 360
```

```
<NA>                    Student
## 361
<NA>             Informal worker
## 362
<NA>                    Student
## 363
<NA>                    Student
## 364
<NA>                    Student
## 365
<NA>                    Student
## 366
Paracetamol/Antibiotics/Other                    Student
## 367
<NA>             Salaried worker
## 368
<NA>             Informal worker
## 369
Paracetamol/Antibiotics                    Student
## 370
Paracetamol/Antibiotics/Other                    Student
## 371                                                          Traditional
meds.                    Student
## 372                                                          Traditional
meds.             Salaried worker
## 373                                             Antibiotics/Other anti-
inflamm.                    Student
## 374                                          Paracetamol/Traditional
meds./Other                    Student
## 375
Other                    Student
## 376
<NA>                    Student
## 377
<NA>                    Student
## 378
<NA>                    Student
## 379
<NA>                    Student
## 380                                 Paracetamol/Antibiotics/Traditional
meds.               Retired/Other
## 381
Paracetamol                 Home-maker
## 382                                                               No
medication                    Trader
## 383                                                          Traditional
meds.                    Retired
## 384                                                          Traditional
meds.                     Trader
## 385
<NA>                    Student
## 386
<NA>             Informal worker
## 387
<NA>             Informal worker
## 388
```

```
Paracetamol             Salaried worker
## 389
<NA>             Informal worker
## 390
<NA>                   Unemployed
## 391
<NA>                   Home-maker
## 392
<NA>                   Home-maker
## 393
Other Informal worker/Unemployed
## 394                                                                  No
response                     Student
## 395
<NA>             Salaried worker
## 396                                                        Traditional
meds.                      Student
## 397                                                        Traditional
meds.                      Student
## 398
Antibiotics                      Student
## 399
Paracetamol             Salaried worker
## 400
<NA>                   Home-maker
## 401
Antibiotics                      Student
## 402
<NA>             Informal worker
## 403
<NA>             Salaried worker
## 404
<NA>             Salaried worker
## 405
<NA>                     Student
## 406
<NA>             Informal worker
## 407
<NA>                      Farmer
## 408                                                              No
medication             Informal worker
## 409
<NA>                      Trader
## 410
<NA>                     Student
## 411
<NA>                     Student
## 412                                                  Other anti-
inflamm.                  Unemployed
## 413
<NA>             Informal worker
## 414
<NA>             Informal worker
## 415
<NA>                   Home-maker
## 416
```

```
<NA>                      Student
## 417
<NA>                      Student
## 418
<NA>                      Student
## 419
<NA>                       Trader
## 420
<NA>                   Home-maker
## 421
<NA>                      Student
## 422
<NA>                      Student
## 423
Paracetamol/Antibiotics                    Home-maker
## 424
<NA>                   Home-maker
## 425
<NA>                      Student
## 426
<NA>                   Home-maker
## 427
<NA>              Informal worker
## 428
<NA>                      Student
## 429
<NA>                      Student
## 430
<NA>                      Student
## 431
<NA>                      Student
## 432
<NA>                      Student
## 433
<NA>                      Student
## 434
<NA>              Informal worker
## 435
<NA>                       Trader
## 436
<NA>                      Student
## 437
<NA>                       Trader
## 438
<NA>                      Student
## 439
<NA>                      Student
## 440
<NA>              Informal worker
## 441
<NA>                      Retired
## 442
<NA>                   Unemployed
## 443
<NA>                      Student
## 444
```

```
<NA>                    Student
## 445                                                              No
medication            Informal worker
## 446                                                   Traditional
meds.                    Student
## 447
<NA>                    Student
## 448                              Paracetamol/Antibiotics/Traditional
meds.            Salaried worker
## 449
<NA>                Unemployed
## 450
<NA>                Home-maker
## 451
<NA>                    Student
## 452
<NA>                    Student
## 453
<NA>            Informal worker
## 454
<NA>            Informal worker
## 455
<NA>            Informal worker
## 456
<NA>                    Student
## 457
<NA>                    Student
## 458                               Paracetamol/Antibiotics/Other anti-
inflamm.                   Student
## 459
<NA>            Informal worker
## 460
<NA>            Informal worker
## 461                           Paracetamol/Hydrocortisone/Traditional
meds.                    Student
## 462
Paracetamol                Home-maker
## 463
Hydrocortisone            Informal worker
## 464
<NA>            Informal worker
## 465
<NA>            Informal worker
## 466
<NA>                    Trader
## 467
Paracetamol/Hydrocortisone                  Student
## 468                              Paracetamol/Antibiotics/Traditional
meds.            Salaried worker
## 469
<NA>                    Student
## 470
Paracetamol/Other            Informal worker
## 471                                                   Traditional
meds.            Informal worker
## 472
```

```
<NA>                     Student
## 473
<NA>                     Student
## 474
Paracetamol    Student/Informal worker
## 475
<NA>                     Student
## 476
<NA>                     Student
## 477
<NA>                  Home-maker
## 478
<NA>                  Home-maker
## 479                                          Paracetamol/Traditional
meds./Other            Informal worker
## 480
<NA>                     Student
## 481
<NA>                     Student
## 482
<NA>            Informal worker
## 483
<NA>                     Student
## 484
<NA>                     Student
## 485                                  Paracetamol/Antibiotics/Traditional
meds.                     Student
## 486
<NA>                     Student
## 487
<NA>                     Student
## 488
<NA>                     Student
## 489
<NA>                      Trader
## 490
Paracetamol/Antibiotics/Other                      Trader
## 491
Paracetamol            Informal worker
## 492
<NA>                  Unemployed
## 493                                                          Traditional
meds.                     Student
## 494
<NA>                      Trader
## 495
Paracetamol/Other                 Unemployed
## 496
<NA>            Informal worker
## 497
<NA>                     Student
## 498                                     Paracetamol/Other anti-
inflamm.            Salaried worker
## 499
Paracetamol/Antibiotics/Other                     Student
## 500
```

```
<NA>                    Student
##  [ reached 'max' / getOption("max.print") -- omitted 471 rows ]
```

Other useful checks in your column strings are: - finding whitespaces (" ") and replace them with ("_") because whitespaces are never desirable in programming. You can reinsert them once you want to plot.

4. Remove the value "1" from the variable `county`.

`str_replace_all`

It is good practice to check for special characters (vector, known in R as `[[:punct:]]`) in your strings. This rigor is especially useful if you are working in another language than English (such as French), which uses accents, because accents are often transformed into special characters by computers.

We will use `str_replace_all` from the {`stringr`} package, which allows you to input a vector of characters that you want to replace by another (commonly to remove them we use the "" character).

The syntax is as follows: `str_replace_all(column, vector_of_characters_to_replace, replacement_character)`

```
yaounde %>%
    mutate(across(where(is.character),
           ~ str_replace_all(.x, "[[:punct:]]","")))
```

```
##                id_ind age age_category    sex   education      occupation
weight_kg height_cm    smoker    pregnant medicated
## 1  BRIQUETERIE0000001  45       45  64 Female  Secondary1 Informal worker
95        169 Nonsmoker          NO       YES
## 2  BRIQUETERIE0000002  55       45  64    Mal University1 Salaried worker
96        185  Exsmoker       <NA>        NO
## 3  BRIQUETERIE0000003  23       15  29      M University1         Student
74        180    Smoker       <NA>       YES
## 4  BRIQUETERIE0020001  20       15  29 Female  Secondary1         Student
70        164 Nonsmoker          NO        NO
## 5  BRIQUETERIE0020002  55       45  64 female    Primary1    TraderFarmer
```

```
67       147 Nonsmoker          NO         NO
## 6  BRIQUETERIE0020003  17       15  29 Female  Secondary1          Student
65       162 Nonsmoker          NO         NO
## 7  BRIQUETERIE0020004  13        May14      F  Secondary1          Student
65       150 Nonsmoker          NO         NO
## 8  BRIQUETERIE0030001  28       15  29   Male  Doctorate1          Student
62       173 Nonsmoker       <NA>         NO
## 9  BRIQUETERIE0030005  16       15  29 Female  Secondary1          Student
70       164 Nonsmoker          NO         NO
## 10 BRIQUETERIE0030002  30       30  44   Male  Secondary1           Trader
73       170 Nonsmoker       <NA>         NO
## 11 BRIQUETERIE0030003  13        May14 Female  Secondary1            Other
56       153 Nonsmoker          NO         NO
## 12 BRIQUETERIE0030004  62       45  64 Female    Primary1       Unemployed
53       157 Nonsmoker NO RESPONSE         NO
## 13 BRIQUETERIE0050001  34       30  44 Female    Primary1           Trader
78       164 Nonsmoker          NO         NO
## 14 BRIQUETERIE0050002  38       30  44 Female    Primary1        Homemaker
58       169 Nonsmoker          NO         NO
## 15 BRIQUETERIE0050003  42       30  44   Male  Secondary1           Trader
71       171  Exsmoker       <NA>         NO
## 16 BRIQUETERIE0050004  28       15  29 Female    Primary1 Informal worker
52       155 Nonsmoker          NO         NO
## 17 BRIQUETERIE0050005  21       15  29 Female    Primary1 Informal worker
56       160 Nonsmoker          NO         NO
## 18 BRIQUETERIE0050006   8        May14 Female    Primary1          Student
26       127 Nonsmoker          NO         NO
## 19 BRIQUETERIE0050007   8        May14   Male    Primary1          Student
29       120 Nonsmoker       <NA>         NO
## 20 BRIQUETERIE0050008   7        May14   Male    Primary1          Student
20       115 Nonsmoker       <NA>         NO
## 21 BRIQUETERIE0050009   9        May14   Male    Primary1          Student
28       134 Nonsmoker       <NA>         NO
## 22 BRIQUETERIE0070002  15       15  29   Male  Secondary1          Student
51       173 Nonsmoker       <NA>         NO
## 23 BRIQUETERIE0050010  31       30  44 Female    Primary1           Trader
75       174 Nonsmoker          NO         NO
## 24 BRIQUETERIE0050011  38       30  44   Male University1 Informal worker
71       175  Exsmoker       <NA>         NO
## 25 BRIQUETERIE0050012   7        May14 Female    Primary1          Student
28       131 Nonsmoker          NO         NO
## 26 BRIQUETERIE0070001  54       45  64   Male University1 Salaried worker
71       180    Smoker       <NA>         NO
## 27 BRIQUETERIE0070003  42       30  44 Female    Primary1        Homemaker
101       175 Nonsmoker          NO         NO
## 28 BRIQUETERIE0070004  18       15  29   Male  Secondary1          Student
64       177 Nonsmoker       <NA>         NO
##    household_with_children  breadwinner
source_of_revenue          has_contact_covid igg_result igm_result
## 1              NO CHILDREN  NO RESPONSE
OTHER       RECENT COVID CONTACT   Negative   Negative
## 2              NO CHILDREN OTHER FAMILY          REGULAR FORMALREGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 3              NO CHILDREN  NO RESPONSE
OTHER           NO COVID CONTACT   Negative   Negative
## 4            WITH CHILDREN       MOTHER                       IRREGULAR
```

```
COMMERCE           NO COVID CONTACT   Positive   Negative
## 5            WITH CHILDREN       MOTHER REGULAR INFORMALIRREGULAR
COMMERCEOTHER           NO COVID CONTACT   Positive   Negative
## 6            WITH CHILDREN  NO RESPONSE                      IRREGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 7            WITH CHILDREN  NO RESPONSE
OTHER            NO COVID CONTACT   Positive   Negative
## 8            WITH CHILDREN  NO RESPONSE      IRREGULAR FORMALIRREGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 9            WITH CHILDREN  NO RESPONSE      IRREGULAR FORMALIRREGULAR
COMMERCE           NO COVID CONTACT   Positive   Positive
## 10           WITH CHILDREN  NO RESPONSE      IRREGULAR FORMALIRREGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 11           WITH CHILDREN  NO RESPONSE      IRREGULAR FORMALIRREGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 12           WITH CHILDREN OTHER FAMILY                        REGULAR
INFORMAL           NO COVID CONTACT   Positive   Negative
## 13           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 14           WITH CHILDREN       MOTHER                        REGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 15           WITH CHILDREN FATHERMOTHER                        REGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 16           WITH CHILDREN       FATHER                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 17           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Negative   Positive
## 18           WITH CHILDREN       FATHER                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 19           WITH CHILDREN       FATHER                        REGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 20           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 21           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Negative   Negative
## 22           WITH CHILDREN       FATHER                          REGULAR
FORMAL          NO COVID CONTACT   Negative   Negative
## 23           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 24           WITH CHILDREN       FATHER                        REGULAR
COMMERCE UNSURE ABOUT COVID CONTACT   Negative   Negative
## 25           WITH CHILDREN  NO RESPONSE                        REGULAR
COMMERCE           NO COVID CONTACT   Positive   Negative
## 26           WITH CHILDREN       FATHER                          REGULAR
FORMAL          NO COVID CONTACT   Negative   Positive
## 27           WITH CHILDREN       FATHER                          REGULAR
FORMAL          NO COVID CONTACT   Positive   Negative
## 28           WITH CHILDREN       FATHER                          REGULAR
FORMAL          NO COVID CONTACT   Negative   Positive
##                                          symptoms    consultation
treatment_combinations             drugsource            hospitalised
## 1                                    Muscle pain            <NA>
Paracetamol       Self or familial                      No
## 2                                    No symptoms            <NA>
<NA>                   <NA>                        No
## 3                                    No symptoms            <NA>
```

```
<NA>                    <NA> Yes but not COVIDlinked
## 4              RhinitisSneezingAnosmia or ageusia            <NA>
Antibiotics       Self or familial                         No
## 5                                    No symptoms            <NA>
<NA>                    <NA>                         No
## 6  FeverCoughRhinitisNausea or vomitingDiarrhoea  Private clinic
ParacetamolAntibiotics             Pharmacist                      No
## 7                                      Sneezing            <NA>
Traditional meds       Self or familial                       No
## 8                                      Headache            <NA>
Paracetamol       Self or familial                         No
## 9                         RhinitisSneezingHeadache            <NA>
Paracetamol       Self or familial                         No
## 10                FeverRhinitisAnosmia or ageusia            <NA>
ParacetamolTraditional meds DoctorSelf or familial                        No
## 11                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 12                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 13                                CoughHeadache             <NA>
ParacetamolAntibiotics       Self or familial                        No
## 14                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 15                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 16                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 17                    FeverCoughRhinitisHeadache Public hospital
ParacetamolAntibioticsOther                  Doctor                       No
## 18                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 19                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 20                                   No symptoms            <NA>
<NA>                    <NA>                         No
## 21                                CoughRhinitis             <NA>
Traditional meds       Self or familial                       No
## 22                                         Fever  Private clinic
ParacetamolHydroxychloroquine             Pharmacist Yes but not COVIDlinked
## 23                                CoughHeadache Public hospital
ParacetamolAntibiotics                 Doctor                       No
## 24                          CoughRhinitisHeadache            <NA>
No medication                    <NA>                        No
## 25                               FeverDiarrhoea            <NA>
No medication                    <NA>                        No
## 26                          CoughRhinitisSneezing            <NA>
No medication                    <NA>                        No
## 27                                 FeverHeadache             <NA>
Paracetamol       Self or familial                         No
## 28                                   No symptoms            <NA>
<NA>                    <NA>             No response
##    sequelae respiration_frequency drug_parac drug_antibio
drug_hydrocortisone drug_other_anti_inflam drug_antiviral drug_chloro
drug_tradn
## 1      <NA>                    NA          1            0
0                      0              0           0          0
```

```
## 2      <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 3      <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 4      <NA>                    NA           0           1
0                          0           0           0           0
## 5      <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 6      <NA>                    NA           1           1
0                          0           0           0           0
## 7      <NA>                    NA           0           0
0                          0           0           0           1
## 8      <NA>                    NA           1           0
0                          0           0           0           0
## 9      <NA>                    NA           1           0
0                          0           0           0           0
## 10     <NA>                    NA           1           0
0                          0           0           0           1
## 11     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 12     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 13     <NA>                    NA           1           1
0                          0           0           0           0
## 14     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 15     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 16     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 17     <NA>                    NA           1           1
0                          0           0           0           0
## 18     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 19     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 20     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
## 21     <NA>                    NA           0           0
0                          0           0           0           1
## 22     <NA>                    NA           1           0
0                          0           0           1           0
## 23     <NA>                    NA           1           1
0                          0           0           0           0
## 24     <NA>                    NA           0           0
0                          0           0           0           0
## 25     <NA>                    NA           0           0
0                          0           0           0           0
## 26     <NA>                    NA           0           0
0                          0           0           0           0
## 27     <NA>                    NA           1           0
0                          0           0           0           0
## 28     <NA>                    NA          NA          NA
NA                         NA          NA          NA          NA
##    drug_oxygen drug_other drug_no_resp drug_none
## 1            0          0            0         0
```

```
## 2            NA          NA            NA          NA
## 3            NA          NA            NA          NA
## 4             0           0             0           0
## 5            NA          NA            NA          NA
## 6             0           0             0           0
## 7             0           0             0           0
## 8             0           0             0           0
## 9             0           0             0           0
## 10            0           0             0           0
## 11           NA          NA            NA          NA
## 12           NA          NA            NA          NA
## 13            0           0             0           0
## 14           NA          NA            NA          NA
## 15           NA          NA            NA          NA
## 16           NA          NA            NA          NA
## 17            0           1             0           0
## 18           NA          NA            NA          NA
## 19           NA          NA            NA          NA
## 20           NA          NA            NA          NA
## 21            0           0             0           0
## 22            0           0             0           0
## 23            0           0             0           0
## 24            0           0             0           1
## 25            0           0             0           1
## 26            0           0             0           1
## 27            0           0             0           0
## 28           NA          NA            NA          NA
##  [ reached 'max' / getOption("max.print") -- omitted 943 rows ]
```

Homogenize all strings throughout the dataset

Here, to easily manipulate all our data in the future, we transform all our strings to lowercase using the `tolower()` function. We select all character type columns with `where(is.character)`

```
yaounde %>%
  mutate(across(where(is.character),
                ~ tolower(.x)))
```

```
##                  id_ind age age_category    sex   education
occupation weight_kg height_cm     smoker    pregnant medicated
## 1  briqueterie_000_0001  45      45 - 64 female  secondary1 informal
worker         95        169 non-smoker          no       yes
## 2  briqueterie_000_0002  55      45 - 64    mal university1 salaried
worker         96        185  ex-smoker        <NA>        no
## 3  briqueterie_000_0003  23      15 - 29       m university1
student         74        180     smoker        <NA>       yes
## 4  briqueterie_002_0001  20      15 - 29 female  secondary1
student         70        164 non-smoker          no        no
## 5  briqueterie_002_0002  55      45 - 64 female    primary1  trader--
farmer         67        147 non-smoker          no        no
## 6  briqueterie_002_0003  17      15 - 29 female  secondary1
```

```
student        65       162 non-smoker          no        no
## 7  briqueterie_002_0004  13       may-14      f  secondary1
student        65       150 non-smoker          no        no
## 8  briqueterie_003_0001  28      15 - 29   male  doctorate1
student        62       173 non-smoker        <NA>        no
## 9  briqueterie_003_0005  16      15 - 29 female  secondary1
student        70       164 non-smoker          no        no
## 10 briqueterie_003_0002  30      30 - 44   male  secondary1
trader         73       170 non-smoker        <NA>        no
## 11 briqueterie_003_0003  13       may-14 female  secondary1
other         56       153 non-smoker          no        no
## 12 briqueterie_003_0004  62      45 - 64 female    primary1
unemployed        53       157 non-smoker no response        no
## 13 briqueterie_005_0001  34      30 - 44 female    primary1
trader         78       164 non-smoker          no        no
## 14 briqueterie_005_0002  38      30 - 44 female    primary1      home-
maker         58       169 non-smoker          no        no
## 15 briqueterie_005_0003  42      30 - 44   male  secondary1
trader         71       171  ex-smoker        <NA>        no
## 16 briqueterie_005_0004  28      15 - 29 female    primary1 informal
worker         52       155 non-smoker          no        no
## 17 briqueterie_005_0005  21      15 - 29 female    primary1 informal
worker         56       160 non-smoker          no        no
## 18 briqueterie_005_0006   8       may-14 female    primary1
student        26       127 non-smoker          no        no
## 19 briqueterie_005_0007   8       may-14   male    primary1
student        29       120 non-smoker        <NA>        no
## 20 briqueterie_005_0008   7       may-14   male    primary1
student        20       115 non-smoker        <NA>        no
## 21 briqueterie_005_0009   9       may-14   male    primary1
student        28       134 non-smoker        <NA>        no
## 22 briqueterie_007_0002  15      15 - 29   male  secondary1
student        51       173 non-smoker        <NA>        no
## 23 briqueterie_005_0010  31      30 - 44 female    primary1
trader         75       174 non-smoker          no        no
## 24 briqueterie_005_0011  38      30 - 44   male university1 informal
worker         71       175  ex-smoker        <NA>        no
## 25 briqueterie_005_0012   7       may-14 female    primary1
student        28       131 non-smoker          no        no
## 26 briqueterie_007_0001  54      45 - 64   male university1 salaried
worker         71       180     smoker        <NA>        no
## 27 briqueterie_007_0003  42      30 - 44 female    primary1      home-
maker        101       175 non-smoker          no        no
## 28 briqueterie_007_0004  18      15 - 29   male  secondary1
student        64       177 non-smoker        <NA>        no
##    household_with_children    breadwinner
source_of_revenue          has_contact_covid igg_result igm_result
## 1               no children    no response
other       recent covid contact   negative   negative
## 2               no children   other family            regular, formal--
regular, commerce           no covid contact   positive   negative
## 3               no children    no response
other           no covid contact   negative   negative
## 4            with children         mother
irregular, commerce           no covid contact   positive   negative
## 5            with children         mother regular, informal--irregular,
```

```
commerce--other           no covid contact   positive   negative
## 6            with children    no response
irregular, commerce           no covid contact   negative   negative
## 7            with children    no response
other           no covid contact   positive   negative
## 8            with children    no response        irregular, formal--
irregular, commerce           no covid contact   negative   negative
## 9            with children    no response        irregular, formal--
irregular, commerce           no covid contact   positive   positive
## 10           with children    no response        irregular, formal--
irregular, commerce           no covid contact   negative   negative
## 11           with children    no response        irregular, formal--
irregular, commerce           no covid contact   positive   negative
## 12           with children   other family
regular, informal           no covid contact   positive   negative
## 13           with children    no response
regular, commerce           no covid contact   positive   negative
## 14           with children         mother
regular, commerce           no covid contact   negative   negative
## 15           with children father--mother
regular, commerce           no covid contact   negative   negative
## 16           with children         father
regular, commerce           no covid contact   positive   negative
## 17           with children    no response
regular, commerce           no covid contact   negative   positive
## 18           with children         father
regular, commerce           no covid contact   positive   negative
## 19           with children         father
regular, commerce           no covid contact   negative   negative
## 20           with children    no response
regular, commerce           no covid contact   positive   negative
## 21           with children    no response
regular, commerce           no covid contact   negative   negative
## 22           with children         father
regular, formal           no covid contact   negative   negative
## 23           with children    no response
regular, commerce           no covid contact   positive   negative
## 24           with children         father
regular, commerce unsure about covid contact   negative   negative
## 25           with children    no response
regular, commerce           no covid contact   positive   negative
## 26           with children         father
regular, formal           no covid contact   negative   positive
## 27           with children         father
regular, formal           no covid contact   positive   negative
## 28           with children         father
regular, formal           no covid contact   negative   positive
##                                                  symptoms    consultation
treatment_combinations               drugsource
## 1                                             muscle pain            <NA>
paracetamol         self or familial
## 2                                             no symptoms            <NA>
<NA>                     <NA>
## 3                                             no symptoms            <NA>
<NA>                     <NA>
## 4                  rhinitis--sneezing--anosmia or ageusia            <NA>
```

```
antibiotics         self or familial
## 5                                          no symptoms            <NA>
<NA>                     <NA>
## 6  fever--cough--rhinitis--nausea or vomiting--diarrhoea  private clinic
paracetamol--antibiotics               pharmacist
## 7                                          sneezing            <NA>
traditional meds.          self or familial
## 8                                          headache            <NA>
paracetamol         self or familial
## 9                          rhinitis--sneezing--headache            <NA>
paracetamol         self or familial
## 10                    fever--rhinitis--anosmia or ageusia            <NA>
paracetamol--traditional meds. doctor--self or familial
## 11                                         no symptoms            <NA>
<NA>                     <NA>
## 12                                         no symptoms            <NA>
<NA>                     <NA>
## 13                                    cough--headache            <NA>
paracetamol--antibiotics         self or familial
## 14                                         no symptoms            <NA>
<NA>                     <NA>
## 15                                         no symptoms            <NA>
<NA>                     <NA>
## 16                                         no symptoms            <NA>
<NA>                     <NA>
## 17                       fever--cough--rhinitis--headache public hospital
paracetamol--antibiotics--other                   doctor
## 18                                         no symptoms            <NA>
<NA>                     <NA>
## 19                                         no symptoms            <NA>
<NA>                     <NA>
## 20                                         no symptoms            <NA>
<NA>                     <NA>
## 21                                    cough--rhinitis            <NA>
traditional meds.          self or familial
## 22                                              fever  private clinic
paracetamol--hydroxychloroquine                pharmacist
## 23                                    cough--headache public hospital
paracetamol--antibiotics                  doctor
## 24                              cough--rhinitis--headache            <NA>
no medication                   <NA>
## 25                                     fever--diarrhoea            <NA>
no medication                   <NA>
## 26                              cough--rhinitis--sneezing            <NA>
no medication                   <NA>
## 27                                     fever--headache            <NA>
paracetamol         self or familial
## 28                                         no symptoms            <NA>
<NA>                     <NA>
##                 hospitalised sequelae respiration_frequency drug_parac
drug_antibio drug_hydrocortisone drug_other_anti_inflam drug_antiviral
## 1                          no     <NA>                    NA           1
0                   0                         0              0
## 2                          no     <NA>                      NA         NA
NA                  NA                        NA             NA
## 3  yes, but not covid-linked     <NA>                      NA         NA
```

```
NA                    NA                          NA              NA
## 4                          no      <NA>                      NA           0
1                     0                           0               0
## 5                          no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 6                          no      <NA>                      NA           1
1                     0                           0               0
## 7                          no      <NA>                      NA           0
0                     0                           0               0
## 8                          no      <NA>                      NA           1
0                     0                           0               0
## 9                          no      <NA>                      NA           1
0                     0                           0               0
## 10                         no      <NA>                      NA           1
0                     0                           0               0
## 11                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 12                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 13                         no      <NA>                      NA           1
1                     0                           0               0
## 14                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 15                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 16                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 17                         no      <NA>                      NA           1
1                     0                           0               0
## 18                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 19                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 20                         no      <NA>                      NA          NA
NA                    NA                          NA              NA
## 21                         no      <NA>                      NA           0
0                     0                           0               0
## 22 yes, but not covid-linked      <NA>                      NA           1
0                     0                           0               0
## 23                         no      <NA>                      NA           1
1                     0                           0               0
## 24                         no      <NA>                      NA           0
0                     0                           0               0
## 25                         no      <NA>                      NA           0
0                     0                           0               0
## 26                         no      <NA>                      NA           0
0                     0                           0               0
## 27                         no      <NA>                      NA           1
0                     0                           0               0
## 28                 no response     <NA>                      NA          NA
NA                    NA                          NA              NA
##    drug_chloro drug_tradn drug_oxygen drug_other drug_no_resp drug_none
## 1            0          0           0          0            0         0
## 2           NA         NA          NA         NA           NA        NA
## 3           NA         NA          NA         NA           NA        NA
## 4            0          0           0          0            0         0
```

```
## 5            NA          NA          NA          NA          NA          NA
## 6             0           0           0           0           0           0
## 7             0           1           0           0           0           0
## 8             0           0           0           0           0           0
## 9             0           0           0           0           0           0
## 10            0           1           0           0           0           0
## 11           NA          NA          NA          NA          NA          NA
## 12           NA          NA          NA          NA          NA          NA
## 13            0           0           0           0           0           0
## 14           NA          NA          NA          NA          NA          NA
## 15           NA          NA          NA          NA          NA          NA
## 16           NA          NA          NA          NA          NA          NA
## 17            0           0           0           1           0           0
## 18           NA          NA          NA          NA          NA          NA
## 19           NA          NA          NA          NA          NA          NA
## 20           NA          NA          NA          NA          NA          NA
## 21            0           1           0           0           0           0
## 22            1           0           0           0           0           0
## 23            0           0           0           0           0           0
## 24            0           0           0           0           0           1
## 25            0           0           0           0           0           1
## 26            0           0           0           0           0           1
## 27            0           0           0           0           0           0
## 28           NA          NA          NA          NA          NA          NA
##  [ reached 'max' / getOption("max.print") -- omitted 943 rows ]
```

5. Transform all the strings in the `typhoid` dataset to lowercase.

Now let's address data types.

8.5 Cleaning data types

Columns containing values that are numbers, factors or logical values (TRUE/FALSE) will only behave as expected if they are correctly classified. As such, you may need to redefine the type or class of your variable.

R has 6 basic data types/classes.

- `character`: strings or individual characters, quoted
- `numeric` : any real numbers (includes decimals)
- `integer`: any integer(s)/whole numbers
- `logical`: variables composed of `TRUE` or `FALSE`

KEY POINT

- `factor`: categorical/qualitative variables
- `Date/POSIXct`: represents calendar dates and times

In addition to the ones listed above, there is also `raw` which will not be discussed in this lesson.

RECAP

In the introduction to this chapter we saw that our data has: - 1 logical variable (`na` column which we removed because all values were `NA`) - 20 character type variables - 15 numeric type variables

Reminder, this is how we observed this:

```
skim(yaounde) %>%
  select(skim_type) %>%
  count(skim_type)
```

WATCH OUT

There are no factor variables when you know the data contains variables such as age, sex: this means we will have quite some work in converting character or numeric variables to factors !

Why is converting categories to factors so important?

KEY POINT

Factor variables are a way for representing categorical variables that can be either numeric or string variables. Factors are stored as integers, and often have labels associated with these unique integers. There are a number of advantages to converting categorical variables to factor variables;

- Factor variables are also essential in rendering plots easily
- Storing string variables as factor variables is a more efficient use of memory
- Factor variables are essential when you want to apply a statistical model to your variable: the model will handle them correctly, assigning the correct number of degrees of freedom

(if this does not ring any bells for you, no worries, we will delve into statistics into later courses: you will be happy to have your variables as factors then!)

Reminder: the functions to convert a variable to a factor are `as.factor()` from {dplyr} or `factor()` from base R

Now, let's move to cleaning our data types !

Converting categories to factor type

Looking at our data, all our variables are categories, omit `age`, `height_cm`, `weight_kg`, and `respiratory_frequency`.

Here, we already have `age_category` as a variable, else it would be important to create an age category variable: it is often useful to have age aggregates to look at data in plots or statistical analysis.

```
yaounde_factors <-
  yaounde %>%
  mutate(across(!c(age, height_cm, weight_kg, respiration_frequency),
                ~ as.factor(.x)))

skim(yaounde_factors) %>%
  select(skim_type) %>%
  count(skim_type)
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## ✖ Column `skim_type` doesn't exist.
```

As a reminder from your data wrangling lesson: we use ~ to indicate that we're supplying an anonymous function and use `.x` to indicate where the variables supplied in `across()` are used.

6. Convert the variables in position 13 to 29 to factor.

Using factors to recode values

All drug related variables are encoded as numeric variables. However, they are categories! They represent if a patient took (encoded by 1) or if they did not take (encoded by 0) that drug.

If you want finer control while creating factors, use the `factor()` function. `as.factor()` should suffice in most cases but use `factor()` when you want to:

- specify levels
- modify labels
- include NA as a level/category
- create ordered factors
- specify order of levels

The syntax is `factor(x,levels, labels = levels )`, where x is the variable in question.

0 and 1 may not be very clear annotations of if a patient took a drug or not: let's transform all the drug variables into factors, while naming their levels with representative "No" (did not take the drug) instead of 0 and "Yes" (took the drug) instead of 1.

```
yaounde %>%
mutate(across(contains("drug"), # selects all variables reflecting a drug
         intake
               ~ factor(.x, # handles the variables one by one
                        levels = c(0:1), # selects the current binary varialbes
                        labels = c("No","Yes")))) %>% # and renames then with
         clearer annotations
  select(contains("drug"))
```

```
##    drugsource drug_parac drug_antibio drug_hydrocortisone
drug_other_anti_inflam drug_antiviral drug_chloro drug_tradn drug_oxygen
drug_other
## 1         <NA>         Yes           No                  No
No             No          No          No          No          No
## 2         <NA>        <NA>         <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 3         <NA>        <NA>         <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 4         <NA>          No          Yes                 No
No             No          No          No          No          No
## 5         <NA>        <NA>         <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
```

```
## 6        <NA>        Yes          Yes                 No
No             No          No          No          No          No
## 7        <NA>         No          No                  No
No             No          No         Yes          No          No
## 8        <NA>        Yes          No                  No
No             No          No          No          No          No
## 9        <NA>        Yes          No                  No
No             No          No          No          No          No
## 10       <NA>        Yes          No                  No
No             No          No         Yes          No          No
## 11       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 12       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 13       <NA>         Yes         Yes                 No
No             No          No          No          No          No
## 14       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 15       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 16       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 17       <NA>         Yes         Yes                 No
No             No          No          No          No        Yes
## 18       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 19       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 20       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 21       <NA>          No          No                  No
No             No          No         Yes          No          No
## 22       <NA>         Yes          No                  No
No             No         Yes          No          No          No
## 23       <NA>         Yes         Yes                 No
No             No          No          No          No          No
## 24       <NA>          No          No                  No
No             No          No          No          No          No
## 25       <NA>          No          No                  No
No             No          No          No          No          No
## 26       <NA>          No          No                  No
No             No          No          No          No          No
## 27       <NA>         Yes          No                  No
No             No          No          No          No          No
## 28       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 29       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 30       <NA>         Yes          No                  No
No             No          No         Yes          No        Yes
## 31       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 32       <NA>        <NA>        <NA>                <NA>
<NA>           <NA>        <NA>        <NA>        <NA>        <NA>
## 33       <NA>          No         Yes                 No
No             No          No         Yes          No        Yes
```

```
## 34       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 35       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 36       <NA>        Yes           No                  No
No             No         No          No         No        Yes
## 37       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 38       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 39       <NA>         No          No                  No
No             No         No         Yes         No         No
## 40       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 41       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 42       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 43       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 44       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 45       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 46       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 47       <NA>        Yes           No                  No
No             No         No         Yes         No         No
## 48       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 49       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 50       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 51       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 52       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 53       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 54       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 55       <NA>         No          No                  No
No             No         No         Yes         No         No
## 56       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 57       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 58       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 59       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 60       <NA>       <NA>          <NA>                <NA>
<NA>           <NA>       <NA>        <NA>       <NA>       <NA>
## 61       <NA>        Yes          Yes                 No
No             No         No         Yes         No         No
```

```
## 62       <NA>         Yes          Yes                  No
No             No          No          No          No          No
## 63       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 64       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 65       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 66       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 67       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 68       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 69       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 70       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 71       <NA>         Yes          Yes                  No
No             No          No          No          No          No
## 72       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 73       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 74       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 75       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 76       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 77       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 78       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 79       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 80       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 81       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 82       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
## 83       <NA>        <NA>         <NA>                 <NA>
<NA>           <NA>        <NA>         <NA>        <NA>         <NA>
##    drug_no_resp drug_none
## 1            No        No
## 2          <NA>      <NA>
## 3          <NA>      <NA>
## 4            No        No
## 5          <NA>      <NA>
## 6            No        No
## 7            No        No
## 8            No        No
## 9            No        No
## 10           No        No
## 11         <NA>      <NA>
```

```
## 12          <NA>       <NA>
## 13            No         No
## 14          <NA>       <NA>
## 15          <NA>       <NA>
## 16          <NA>       <NA>
## 17            No         No
## 18          <NA>       <NA>
## 19          <NA>       <NA>
## 20          <NA>       <NA>
## 21            No         No
## 22            No         No
## 23            No         No
## 24            No        Yes
## 25            No        Yes
## 26            No        Yes
## 27            No         No
## 28          <NA>       <NA>
## 29          <NA>       <NA>
## 30            No         No
## 31          <NA>       <NA>
## 32          <NA>       <NA>
## 33            No         No
## 34          <NA>       <NA>
## 35          <NA>       <NA>
## 36            No         No
## 37          <NA>       <NA>
## 38          <NA>       <NA>
## 39            No         No
## 40          <NA>       <NA>
## 41          <NA>       <NA>
## 42          <NA>       <NA>
## 43          <NA>       <NA>
## 44          <NA>       <NA>
## 45          <NA>       <NA>
## 46          <NA>       <NA>
## 47            No         No
## 48          <NA>       <NA>
## 49          <NA>       <NA>
## 50          <NA>       <NA>
## 51          <NA>       <NA>
## 52          <NA>       <NA>
## 53          <NA>       <NA>
## 54          <NA>       <NA>
## 55            No         No
## 56          <NA>       <NA>
## 57          <NA>       <NA>
## 58          <NA>       <NA>
## 59          <NA>       <NA>
## 60          <NA>       <NA>
## 61            No         No
## 62            No         No
## 63          <NA>       <NA>
## 64          <NA>       <NA>
## 65          <NA>       <NA>
## 66          <NA>       <NA>
## 67          <NA>       <NA>
```

```
## 68          <NA>       <NA>
## 69          <NA>       <NA>
## 70          <NA>       <NA>
## 71            No         No
## 72          <NA>       <NA>
## 73          <NA>       <NA>
## 74          <NA>       <NA>
## 75          <NA>       <NA>
## 76          <NA>       <NA>
## 77          <NA>       <NA>
## 78          <NA>       <NA>
## 79          <NA>       <NA>
## 80          <NA>       <NA>
## 81          <NA>       <NA>
## 82          <NA>       <NA>
## 83          <NA>       <NA>
##  [ reached 'max' / getOption("max.print") -- omitted 888 rows ]
```

For some manipulations, it could be better to have the encoding be binary (0 or 1), so this is an example data transformation, it's not imperative to do so.

7. Recode the variable `sex` to factor such that: `0 = female` and `1 = male`.

8.6 Back to inconsistencies

Numeric inconsistencies

For numerical values, it is important to think about our variables and anticipate what errors would cause errors later on.

A few examples: - an abornmal occurence would be having negative values for variables that can only be positive (age, height, weight) - a problematic data recording includes have `Inf` values for numbers

Let's see how to address both of these issues.

We will keep `NA` values and see how to address these more specifically in the next lesson

Ensuring non-negative variables

Here we will check that all our numeric variables are either `NA` (have not been entered), with the `is.na(.x)` condition, or are coherent i.e. are all positive values, with the `.x >= 0` condition.

```
nrow(yaounde)
```

```
## [1] 971
```

```
yaounde_positive_numeric <-
  yaounde_factors %>%
  filter(across(where(is.numeric),
                ~ (.x >=0 | is.na(.x)) ))

nrow(yaounde_positive_numeric)
```

```
## [1] 971
```

From this filter, we see that none of our numeric variables had erroneous negative values: yay! It is always important to check this because you do not know until you check.

Ensuring finite variables

Here we will check that all our numeric variables are either `NA` (have not been entered), with the `is.na(.x)` condition, or are a finite number, with the `is.finite(.x)` condition.

```
nrow(yaounde)
```

```
## [1] 971
```

```
yaounde_finite_numeric <-
  yaounde_factors %>%
  filter(across(where(is.numeric),
                ~ (is.finite(.x) | is.na(.x)) ))

nrow(yaounde_finite_numeric)
```

```
## [1] 971
```

Once again, there are no strange values, such as `Inf`, in our dataset. Good that we checked.

8. Find the number of rows that are either `NA` or negative in the `typhoid` dataset.

8.7 Putting it all together

Time to bring it all together, as you would to do a subsequent analysis.

```
# Let's look with how many rows we started
nrow(yaounde)
```

```
## [1] 971
```

```
yaounde_content_cleaned <-
  yaounde %>%

  # STRING INCONSISTENCIES CORRECTIONS
  # Recode the age categories' strings
  mutate(age_category = recode(age_category,`May-14` = "5-14")) %>%
  # Recode the sex categories' strings
  mutate(sex = recode(sex, `F` = "Female",
                      `female` = "Female",
                      `M` = "Male",
                      `Mal` = "Male")) %>%
  # Change variable annotation for the occupation column
  mutate(occupation = replace(occupation, age < 18 , "Child")) %>%
  # Ensure in all columns that no strings are polluted by special characters
  mutate(across(where(is.character),
                ~ str_replace_all(.x, "[[:punct:]]",""))) %>%
  # Correct the education column's strings
  mutate(education = gsub("1","",education)) %>%
  # Homogenize all strings within all columns are similar: lowercase
  mutate(across(where(is.character),
                ~ tolower(.x))) %>%

  # CHANGE TYPE
  # Transform all categorical variables into factors
  mutate(across(!c(age, height_cm, weight_kg, respiration_frequency),
              ~ as.factor(.x))) %>%
  # Make the drug treatment columns more readable to an external user
  mutate(across(contains("drug"),
              ~ factor(.x,
```

```
                        levels = c(0:1),
                        labels = c("No","Yes")))) %>%
  # NUMERIC INCONSISTENCIES CORRECTIONS
  # Check all numerical variables are positive (because in our case they
        should not be negative)
  filter(across(where(is.numeric),
                          ~ (.x >=0 | is.na(.x)) )) %>%
  # Check all numerical variables are finite
  filter(across(where(is.numeric),
                  ~ (is.finite(.x) | is.na(.x)) ))
# Let's see check we didn't loose any rows from cleaning !
# We are cleaning inside the rows so we expect to have THE SAME NUMBER of rows
        before and after
nrow(yaounde_content_cleaned)
```

```
## [1] 971
```

Let's now save our cleaned dataset !

```
write_csv(yaounde_content_cleaned,

        here::here('ch02_data_cleaning_pipeline/data/yaounde_content_cleaned.csv
```

8.8 Wrapping up

In this chapter, we learnt how to fix inconsistencies and structural errors in the variables. However, there is still one more step in the data cleaning pipeline and that is; **dealing with missingness in the data.** ## Contributors {.unlisted .unnumbered}

The following team members contributed to this lesson:

LAURE VANCAUWENBERGHE
Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Horst, A. (2021). *Dplyr-learnr*. https://github.com/allisonhorst/dplyr-learnr (Original work published 2020)

Artwork was adapted from:

- Horst, A. (2021). *R & stats illustrations by Allison Horst*. https://github.com/allisonhorst/stats-illustrations (Original work published 2018)

9

Data cleaning: missing values

9.1 Learning objectives

1. You explore `NA` values using `is.na()` to test for missing values.
2. You will learn that some `NA` values are normal with the example of gender-specific variables
3. You will replace missing values encoded differently with `NA`s using `na_if()`
4. You will use `replace_na()` to replace `NA` with another value.

9.2 Intro to the lesson

The most common problem related to data cleaning is handling missing values. It can be difficult to perform statistical analysis on data where one or more values in the data are missing. In this lesson, we will explore how to handle missing values.

Too many missing variables: small sample size of complete and hence **a bias** in analysis !

The smaller the dataset, the bigger the bias problem !

9.3 Our data

The data we are cleaning is the data from the COVID-19 serological survey conducted in Yaounde, Cameroon.

We will use the version of the dataset with: - standardised column names - no empty columns and rows - no duplicates - cleaned column content and types

```
#data frame
yaounde <-
        read.csv(here::here('ch02_data_cleaning_pipeline/data/yaounde_cleaned.cs
```

9.4 Identifying missing values

We can identify the number of missing values for each variable from the `n_missing` attribute of the skim output.

```
yaounde %>%
  skim() %>%
  select(skim_variable,n_missing, complete_rate) %>%
  arrange(desc(n_missing))
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## ✖ Column `skim_variable` doesn't exist.
```

1. Identify the number of missing values for each variable in the `typhoid` dataset.

9.5 What to do with variables with missing data ?

Drop the variables

Here we define a vector of columns that are less than 10% complete which we will simply drop altogether from further analysis: they cannot supply unbiased information.

Looking above, we see that these columns are `drugsource`, `consultation`,`sequelae`, and `respiration_frequency`. So we expect to remove 4 columns.

We check that we have correctly dropped these 4 columns by printing `ncol()` before and after our manipulation.

```
columns_with_too_many_missing <-
  yaounde %>%
```

```
  filter(complete_rate < 0.1) %>%         select(skim_variable) %>%
  pull()
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## ✖ Column `skim_variable` doesn't exist.
```

```
ncol(yaounde)
```

```
## [1] 35
```

```
yaounde_dropped_incomplete <-
  yaounde %>%
  select(-columns_with_too_many_missing)

ncol(yaounde_dropped_incomplete)
```

```
## [1] 31
```

```
yaounde_dropped_incomplete %>%
  skim() %>%
  select(skim_variable,n_missing, complete_rate) %>%
  arrange(desc(n_missing))
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## ✖ Column `skim_variable` doesn't exist.
```

Make subsets where the variable has complete information

Your go-to for checking if a column or a row is `NA` in a dataframe is the `is.na()` function.

You may want to keep only patients that have complete drug treatment information.

```
nrow(yaounde)
```

```
## [1] 971
```

```
yaounde_subset_complete_drugs <-
  # We must use the version of the dataset without overly incomplete
        variables, else the variable drugsource will bias our subsetting
  yaounde_dropped_incomplete %>%
  filter(across(contains("drug"),
                ~!is.na(.x)))

nrow(yaounde_subset_complete_drugs)
```

```
## [1] 295
```

The number of patients we can consider for further analysis drops from 971 to 295. Our sample size is reduced but we can still perform further statistical analysis as long as we are careful of checking for biases in this population. We should, for example, plot key variables such as age, BMI, etc. to check that their histograms are not biased.

Some missing data makes sense! The example of gender specific data

As we know, men cannot be pregnant. So having a 56% incompleteness for the variable indicating pregnancy could just reflect the proportion of men in the study, not the percentage of missing data!

Let's look further and contextualize this variable!

```
yaounde %>%
  filter(sex=="female") %>%
  count(pregnant)
```

We see that we have 549 out of 971 patients that are women.

How complete are our records of pregnancy, for women? We see that we only have 4 female patients missing information `NA` on their pregnancy BUT 57 women are also categorized as having given `no response`. In our analysis, this comes down to the same value as `NA` ("we do not know").

We should: - Ask the data collectors if this "no reponse" can mean something more than missing information - Convert the `no response` to `NA`

We can convert a value to `NA` using the `na_if()` function within `mutate()`

The syntax is: `mutate(column = na_if(column,`

value_to_convert_to_NA))

```
yaounde %>%
  filter(sex=="female") %>%
  mutate(pregnant = na_if(pregnant, "no response")) %>%
  count(pregnant)
```

Then we can do two things : - Save this subset dataset for women and work with it when we want to investigate female related inquiries - Change the global dataset to differenciate these two types of `NA`: for men it means "Non applicable" and for a women it is a true `NA` (i.e. missing data)

Let's edit the entire dataset, to keep all our data together.

```
yaounde %>%
  mutate(pregnant = na_if(pregnant, "no response")) %>%
  # by default, all other variables are set to NA
  mutate(pregnant= case_when(sex=="female" & !is.na(pregnant) ~ pregnant,
                             sex=="male" ~ "non applicable")) %>%
  count(pregnant)
```

It's always good to check what we did so let's verify that the number of men is the same as the number of `non applicable`: 422.

```
nrow(yaounde %>% filter(sex=="male"))
```

```
## [1] 422
```

All good !

This way, we are also ensuring data correctness: if the sex of the patient is male, we declare it is impossible for there to be pregnancy information.

Be wary of hidden `NA` values !

We saw for the `pregnant` variable that we had "hidden" `NA` values entered as `no response`. There is the same occurence in the `medicated` variable.

```
yaounde %>%
  count(medicated)
```

Now let's correct it !

```
yaounde %>%
  mutate(medicated = na_if(medicated, "no response")) %>%
  count(medicated)
```

When we put it all together, we can use `across` to handle both columns (`pregnant` and `medicated`) at the same time.

Replace with values / Impute missing data

We will have a complex analysis on this later using MICE. Stay tuned!

In this dataset, `smoker` has 2 missing values! let's look into it.

```
yaounde %>%
  skim() %>%
  select(skim_variable,n_missing, complete_rate) %>%
  filter(skim_variable=="smoker")
```

```
## Error in `select()`:
## ! Can't subset columns that don't exist.
## ✖ Column `skim_variable` doesn't exist.
```

Let's have a look:

```
yaounde %>%
  count(smoker)
```

Imagine you wanted to make a graph and wanted to recode those 2 `NA` to a category name that makes more sense to the general public.

`NA` are of common use for programmers but for a non-coder, general public, they may not even know what an "NA" is. Make sure to always rename them with "missing value" or "unknown" when presenting your data.

The function `replace_na()` allows to substitute all `NA` values with another value (string or numeric). It is combined to `mutate()`

The syntax is as follows:

```
mutate(column = replace_na(column,
replacement_value_string_or_numeric))
```

```
yaounde %>%
  mutate(smoker = replace_na(smoker, "missing information")) %>%
  count(smoker)
```

`na_if` converts:

a value (numeric/character) -> `NA`.

`replace_na` converts:

`NA` -> a value (numeric/character)

9.6 Global overview

As a last step, let's look how many of the variables (`ncol(yaounde)`) collected have a complete information (`complete_rate==1`).

```
ncol(yaounde)
```

```
## [1] 35
```

```
ncol(yaounde %>%
       skim() %>%
       filter(complete_rate==1))
```

```
## Error in (function (cond) : error in evaluating the argument 'x' in
selecting a method for function 'ncol': Problem while computing `..1 =
complete_rate == 1`.
```

```
## Caused by error in `complete_rate == 1`:
## ! comparison (1) is possible only for atomic and list types
```

A bit more than half our variables have complete information. Seeing that these include essential demographic information such as `age`, `sex` etc, this is pretty satisfying. We can perform an analysis using this dataset.

9.7 Putting it all together

```
yaounde_missing_handled <-
  yaounde %>%
  # remove unusable data columns defined above
  select(-columns_with_too_many_missing) %>%
  # handle hidden NA
  mutate(across(c(pregnant,medicated),
                ~ na_if(.x, "no response"))) %>%
  # handle normal missing values by redefining them
  mutate(pregnant= case_when(sex=="female" & !is.na(pregnant) ~ pregnant,
                             sex=="male" ~ "non applicable"))
```

Missing values were the LAST step !!

Let's now save our totally cleaned dataset, ready for plotting or statistical analysis: we made it, **BRAVO** !

```
write_csv(yaounde_missing_handled,
          here::here('ch02_data_cleaning_pipeline/data/yaounde_clean.csv'))
```

Contributors

The following team members contributed to this lesson:

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network

A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- Horst, A. (2021). *Dplyr-learnr*. https://github.com/allisonhorst/dplyr-learnr (Original work published 2020)
- *Subset columns using their names and types—Select*. (n.d.). Retrieved 31 December 2021, from https://dplyr.tidyverse.org/reference/select.html

Artwork was adapted from:

- Horst, A. (2021). *R & stats illustrations by Allison Horst*. https://github.com/allisonhorst/stats-illustrations (Original work published 2018)

10

Functions: introduction

10.1 Learning objectives

1. You will learn how to create your own functions.
2. You will learn the importance of creating your own functions as you improve your career as a developing data scientist.
3. You will learn how to write the body of a function.
4. You will learn how to design and handle the arguments (parameters) of a function.

10.2 Introduction to the dataset

In this lesson, we will continue to to use the COVID-19 dataset containing the results from a serological survey conducted in Yaounde, Cameroon in late 2020.

```
yaounde <- read_csv(here("ch04_functions/data/yaounde_data.csv"))
yaounde
```

10.3 Introducing functions

R language is based mainly on using objects and functions. Quoting John Chambers, everything that "exists" in a R environment is an object, and everything that "happens" is a function.

So far, we have already encountered and used a large number of functions. Functions can be quite simple (such as `sum()`, `max()`, `mean()`), or really complex (such as what we are seeing in the `tidyverse` package like `mutate()`, and even more so functions related to statistical analysis which you will discover in further courses).

@ILLUSTRATE:

The basic principle behind a function is that it needs to receive as input one or more arguments (or parameters), to perform a certain number of actions so as to return a result. Functions are programmed in a way that the user does not see its code running; all calculations and actions are usually done in the background. Finally, the function will return the desired result, which can - but not always - be accompanied by a message or even an error. More on these later.

10.4 Why should I bother writing functions?

Writing functions is a very important task to any data scientist, from beginners to seasoned coders. Functions not only make you more experienced with the R language, but it allows you to speed up your work and even reduce the chance of making errors while coding. You will be able to automate common and repetitive tasks in your data analysis flow (such as calculating epidemiological indicators, for example). This is very important because it will allow you to drop the dreaded copy-and-paste routine that plagues many good scripts (which can easily turn into hundreds or thousands of copied redundant lines).

@ILLUSTRATE

The main advantages of writing functions over copy-and-pasting are:

- You can give a function an evocative name that makes your code easier to understand.
- As requirements change, you only need to update code in one place, instead of many.
- You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).

10.5 You convinced me, when do we start to writing our own functions?

In a few moments. But, before starting with our coding, you should know that writing your own functions is a lifetime journey. Even experienced programmers learn new techniques and better ways of approaching old problems. This happens because as we grow more experienced in our path as data scientists, we are exposed to new problems and solving them gives us new insights in turn. So, don't worry about mastering every little detail, because it is just not possible to list them all. This lesson will provide you with some advice that you can learn and start applying immediately.

Also, don't worry if things seem complicated. As said above, time and experience (and a lot of testing!) will make you write good and useful functions in no time!

- So far, what have you understood about writing your own functions in R?

  A. To be able to automate tasks that are needed in the script and minimize the chance of erring when we need to copy-and-paste chunks of code multiple times.

  B. Functions are a less effective way of repeating chunks of code that we can easily copy-and-paste in our script.

  C. It is easier to spot mistakes and typos when we are copying-and-pasting repeated chunks of code, since we are familiar with it.

10.6 Basics of a function

Here is an example of a very simple function that only adds 2 to a number:

```
sum_2 <- function(x) {
  result <- x + 2
  return(result)
}
```

If you execute this code, you will create a function named `sum_2`, which can be used directly in a script or in the console:

```
sum_2(12)
```

```
## [1] 14
```

Before going further, let's break down the structure of this first function step by step. First, a function is created using the statement `function`. This is followed by a pair of parentheses and a pair of braces.

```
function() {

  }
```

Inside the parentheses, we indicate the arguments of the function, those that must be passed to it when we call it. Our function only takes one argument, which we have arbitrarily decided to name `x`.

```
function(x) {

  }
```

The braces include a series of R statements that make up the **body** of the function. This is the code that will be executed when our function is called. The arguments passed to it must be used here inside the body of the function. In our function `sum_2`, the first line takes the value of the argument x, adds 2 to it, and stores the result in a new object `result`.

```
function(x) {
  result <- x + 2
}
```

Now we want that our function return what is calculated inside its body. This is achieved via the instruction `return` to which we pass the value to return (in our case, the object `result`).

```
function(x) {
  result <- x + 2
  return(result)
}
```

Finally, in order for our function to be called and used, we need to give it a name. This is the same as storing a value in an object. Here we store it in an object named `sum_2`.

```
sum_2 <- function(x) {
  result <- x + 2
  return(result)
}
```

With our function, we have therefore created a new object in our environment called `sum_2`, of type `function`. This new function takes a single argument `x`, calculates the value `x + 2` and returns that result. It is used by typing its name followed by the value of its argument in parentheses, in the same manner that we have done so far in our course. For example:

```
sum_2(12)
```

```
## [1] 14
```

Or:

```
sum_2(x = 12)
```

```
## [1] 14
```

You can also assign the value to an object and pass it to the function. This is what the pros do in their scripts, since it avoids what is called "hardcoding":

```
value <- 12

sum_2(value)
```

```
## [1] 14
```

Why is hardcoding bad?

Hardcoding is a common bad practice of inputing data directly into the code, as opposed to obtaining the data from an external source. Whenever possible, we should avoid hardcoding our scripts in order to prevent errors. In very long scripts, the act of hardcoding might cause unnexpected errors (especially if the user is a copy-and-paste addict). This is analogous to our initial examples of why you should write your own functions: if copying-and-pasting chunks of repeated code is by itself prone to errors, imagine copying-and-pasting chunks of code while manually changing some values (hardcoding). This is a review nightmare. In our example above, we avoided hardcoding by assigning the value to an object, which was then passed to the `sum_2` function.

Note that if `x` is a vector, we can also call our function by passing `x` as a vector into the argument. Our vector will be a sequence of 10 up to 50, with jumps of 10 between (10, 20, 30, 40 and 50). This is achieved by the `seq` function in R base.

```
vector <- seq(from = 10, to = 50, by = 10)

sum_2(vector)
```

```
## [1] 12 22 32 42 52
```

This is specially important when working with datasets, since we can pull a variable out of the dataset to apply a function. Coming back to our dataset! Let's apply the function we have just created to the `age` variable of the Yaounde dataset (i.e. adding +2 to the age collected for each participant). We will do this only for the first 10 observations (notice the [1:10], used to select observations 1 through 10).

```
age <- yaounde$age[1:10]

sum_2(age)
```

```
##  [1] 47 57 25 22 57 19 15 30 32 15
```

1. Create a simple function called `age_months` that transforms age in years to age in months. (With the hint being: there are 12 months per year, think multiplication.)

2. Now, apply your function to the yaounde dataset's age variable (i.e. convert the age in years of all participants to an age in months). You should submit a vector.

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU
Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. "Introduction à R et au tidyverse." Accessed May 23, 2022. https://juba.github.io/tidyverse
- Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

11

Functions: multiple arguments

11.1 Learning objectives

1. You will learn how to create more complex functions, with two or more arguments.
2. You will learn how arguments are called in a R function.

11.2 Introduction to the dataset

In this lesson, we will continue to to use the COVID-19 dataset containing the results from a serological survey conducted in Yaounde, Cameroon in late 2020.

```
yaounde <- read_csv(here("ch04_functions/data/yaounde_data.csv"))

yaounde
```

11.3 Functions with more than one argument and definition of arguments

It is definitely possible to write more complex functions with multiple arguments. For instance:

```
sum_y <- function(x, y) {
  result <- x + y
  return(result)
}

sum_y(1, 12)
```

```
## [1] 13
```

In this example, we defined that our function should have an `x` and a `y` arguments. If you try to use the function without passing one of the arguments, it will result in an error.

This happens because whenever we create a function in which we do not define default values for the arguments, they are mandatory. For example:

```
sum_y(1)
```

```
## Error in sum_y(1): argument "y" is missing, with no default
```

This generates an error, since both `x` and `y` arguments do not have any default values attributed to them.

Default values for the arguments of a function

You can define **default values** for the arguments of your function. In this manner, if a function is **called** without **attributing a value to an argument**, then this argument takes its default value.

See the example below, where `y` is given the default value of `0`.

```
sum_y <- function(x, y = 0) {
  result <- x + y
  return(result)
}

sum_y(1)
```

```
## [1] 1
```

You will notice that we defined the default value for `y` as `0`. So, using `sum_y(1)` will not return any error message. Thus, `x` is a mandatory argument and `y` is an optional argument in our last example.

If we would like an argument to be optional, but we don't want it to necessarily have a default value, we can assign the default value to `NULL`. Just pay attention that `NULL` values are of the class `NULL` (just type in the console `class(NULL)`). Remember that you can't mix different types of data! For instance, in our function `sum_y`, you can't make the operation `sum_y(1, NULL)`. Try it for yourself.

The order of the arguments

If you paid attention so far, you might have noticed that one way of passing arguments to a function is simply by inputing them by position. For instance:

```
simple_function <- function(arg1, arg2, arg3) {
  cat(arg1, arg2, arg3)
}

x <- "Hello!"

simple_function(x, 12, TRUE)
```

```
## Hello! 12 TRUE
```

In this case, `arg1` will be `x` (which, in turn, received the word "Hello!"), `arg2` will be `12`, and `arg3` will be `TRUE`.

When you pass the arguments by name, you can specify them in any order you want:

```
simple_function(arg1 = x, arg3 = TRUE, arg2 = 12)
```

```
## Hello! 12 TRUE
```

And we can obviously mix pass by position and pass by name, and even change their order:

```
simple_function(12, arg1 = x, arg3 = TRUE)
```

```
## Hello! 12 TRUE
```

You can give several arguments to a function **by position** `function(value1,value2,value3)` or **by name** `function(arg1=value1, arg2=value2, arg3=value3)`.

If you give them via **name** then **the order does not matter**: `function(arg3=value3, arg1=value1, arg2=value2)`

BUT

If you give them via **position** only, then **order DOES matter**: `function(value2, value3, value1)` will be intaken by your function

as `function(arg1=value2, arg2=value3, arg3=value1)`. It's a mess and can lead to errors !

Often, the first argument in a R function is the data on which it will work, while the following arguments are parameters that will modify its behavior. For example, the function `median` accepts as its first argument `x`, which is a vector of numerical values, and then an argument `na.rm` that will change the way it calculates the median of the values of `x`. Remember that `na.rm` is a logical argument of some functions that strip the NA values before doing the calculations!

1. Create a function called `BMI` that calculates the Body Mass Index (BMI) of one or more persons. Remember that BMI is calculated as weight in kg divided by the squared height (in meters). This is obviously a function that needs two mandatory arguments: weight and height.

Tip: check the dataset before writing your function. (OR look back at what you learnt with `mutate` in *Introductory Data Analysis with R* - Data wrangling with dplyr chapter)

2. Use the `yaounde` dataset to compute the BMI of the patients in the study.

11.4 More complex functions

Functions can become as complex as you want. Just make sure to not hardcode and to test everything.

The following is a much more complex version of the BMI function, which allows to compute a table with the counts of subjects in each category (BMI allows you to determine if the person is obese or not).

If the example is scary to you now, **don't worry** ! by the end of this chapter you will feel at ease with functions like these.

You will notice the use of a function from R base called `cut`. We have seen it in the **conditional statements** lessons. Remember that you can always learn more about a function by using the help function. Type `?cut` or `help(cut)` in the console to learn more about it.

Also, you will notice that we introduced a conditional statement in our function. We used the `if` function to make an internal test of the argument `return_table`. According to the `if` chain, if `return_table = TRUE`, the function must return the internal object `BMI_table`. Otherwise (as specified by the `else` statement) it will return the vector of BMI values `x`.

```
BMI <- function(weight, height, return_table = FALSE) {

  x <- weight / (height ^ 2)

  BMI_classes <- cut(x,
                     breaks = c(-Inf, 18.5, 24.9, 29.9, 34.9, 39.9, Inf),
                     labels = c("Underweight", "Normal weight", "Overweight",
        "Obesity class I", "Obesity class II", "Obesity class III"))

  BMI_table <- table(BMI_classes)

  if (return_table == TRUE) {
    return(BMI_table)
  } else {
      return(x)
  }
}
```

Let's check our new BMI functions when defining `return_table = FALSE`. We will use data from the `yaounde` dataset. Notice that we are converting the height from centimeters to meters by dividing the vector by 100. Also, since the final result will be a vector of 971 numbers, we will limit the output to just the first 10 observations.

```
ya_w <- yaounde$weight_kg

ya_h <- yaounde$height_cm / 100

bmi_result1 <- BMI(weight = ya_w, height = ya_h, return_table = FALSE)

bmi_result1[1:10]
```

```
##  [1] 33.26214 28.04967 22.83951 26.02617 31.00560 24.76757 28.88889
20.71569 25.25952 23.92242
```

Now, let's try the BMI functions defining `return_table = TRUE`. Now, we have a beautiful table with the counts for each BMI category, just like we programmed in our function.

```
bmi_result2 <- BMI(weight = ya_w, height = ya_h, return_table = TRUE)

bmi_result2
```

```
## BMI_classes
##       Underweight     Normal weight        Overweight   Obesity class I
Obesity class II Obesity class III
##               161               392               253               104
37                24
```

Note this data set has de column `age_category`. This column is really useful to have known about the behavior of the other columns of the data set in different age groups. But, sometimes, a data set provides a column with age but not a column as `age_category`, or, the range provided in the age category can not be useful for you. For this reason, it's interesting to know how to create this column given a data set with an age column. So, let's create a function to do this! To create this function, follow the instructions below:

The function must have four parameters:

- `data`: That receives a data frame that should have a column with age.
- `age_column`: Name of the age column in the data parameter.
- `age_range`: vector with the range of values will be used in the breaks of the `cut` function
- `age_name`: vector with the name of the age groups that will be used in the labels of the `cut` function

The function must return an updated version of the data frame with a new column called `age_group`.

Tip: If you fill the parameters `age_range` and `age_name` as:

- `age_range = c(5,15,30,45,65, Inf)`

- `age_name = c('5 - 14', '15 - 29', '30 - 44', '45 - 64', '65 +')`.

the `age_group` column returned should be equal to the `age_category` column.

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. “Introduction à R et au tidyverse.” Accessed May 23, 2022. https://juba.github.io/tidyverse

Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

12

Functions: naming and commenting

12.1 Learning objectives

1. You will learn the conventions for naming functions.
2. You will learn the importance of commenting your functions (towards reproducible and easily understandable code!).

12.2 General IMPORTANT advice when writing functions (part 1)

Before delving deeper into writing functions using conditional execution, there are a few points that should be said regarding functions.

Naming

Always remember that functions communicate with the computer, but they also need to be comprehensible for humans to read (in the name of reproducible and collaborative programming). The name of a function should clearly state what it does while being short. Ideally, you want the name of your function to be short and informative, but this is not always possible, since it is a hard exercise.

Remember always: **it's better to be clear than short**. Remember that RStudio's autocomplete functionality makes it easy to type long names quickly.

As said by Hadley Wickham, **function names should be verbs, and arguments should be nouns**.

Some exceptions to this are that nouns are ok if the function computes a very well known noun (i.e. `std_error()` is better than `compute_std_error()`). Also, it is ok if you are trying to access properties of an object (i.e. `coef()` is better than

SIDE NOTE

`get_coefficients()`). Use your best judgement and don't be afraid to rename a function if you figure out a better name later.

```
# Too short
f()

# Not a verb, or descriptive
a_great_function()

# Long, but clear
convert_utm()
age_months()
```

WATCH OUT

Maintain consistency when naming your functions. This will be extremely useful if you manage to right your own R package in the future, and also if you share your functions with other colleagues. There aren't any rules written in stone for this. In the R universe, it is common to use **"snake_case" naming** (where each word is lower case and separated by an underscore), or **camelCase**.

```
# Don't do mix naming styles: choose EITHER camelCase or snake_case
convert_utm()
ageMonths()
```

Also, try to have consistent names and arguments if you write functions that do similar things. A good practice is to use a common suffix in these cases, since RStudio's autocomplete will show you other functions of the family

```
# Do this
convert_utm()
convert_wgs84()

# Don't do this
utm_convert()
wgs84_convert()
```

WATCH OUT

Avoid at all costs overriding existing functions and variables, especially from base R.

```
# Never ever do this
mean <- function(x) sum(x)
c <- function(x) print(x)
```

So far, what have you understood about naming your own functions in R? Type T (True) or F (False) to the sentences below:

1. It's better for the function's name to be shorter than clear.
2. It's recommended that function names should be verbs and arguments should be nouns.
3. Try to adopt an standard when naming your functions, separating the words by _ or using uppercase. For example, use `compute_utm()` or `computeUtm()`.
4. It's recommended to override existing functions in the base R as `mean()`.

Commenting

Finally, another good practice is to always comment your code. If you are not doing this already, make this a priority. One of the major advantages of commenting code is that you assure reproducibility not only for your future you but also other colleagues. Use and abuse of the # in your code to explain what you are doing and trying to achieve.

First, let's recap the expanded BMI function that we wrote in last lesson

```
BMI <- function(weight, height, return_table = FALSE) {

  x <- weight / (height ^ 2)

  BMI_classes <- cut(x,
                     breaks = c(-Inf, 18.5, 24.9, 29.9, 34.9, 39.9, Inf),
                     labels = c("Underweight", "Normal weight", "Overweight",
        "Obesity class I", "Obesity class II", "Obesity class III"))

  BMI_table <- table(BMI_classes)

  if (return_table == TRUE) {
    return(BMI_table)
  } else {
      return(x)
  }
}
```

Now, let's see an example of useful comments that will surely improve its documentation and help other users (and even your future you):

```
BMI <- function(weight, height, return_table = FALSE) {
  # Calculating the BMI for each individual
  # weight should be in kilograms
  # height should be in meters
  x <- weight / (height ^ 2)
  # creating categories from the BMI values
  BMI_classes <- cut(x,
                     # creating breaks based on the accepted BMI thresholds
                     breaks = c(-Inf, 18.5, 24.9, 29.9, 34.9, 39.9, Inf),
                     # using official classification to create categories
                     labels = c("Underweight", "Normal weight", "Overweight",
        "Obesity class I", "Obesity class II", "Obesity class III"))
  # creating a table that counts the observations in each BMI category
  BMI_table <- table(BMI_classes)
  # conditional statement; should the function return the BMI_table or the
        vector of numerical BMI values?
  if (return_table == TRUE) {
    return(BMI_table)
  } else {
      return(x)
  }
}
```

This makes the function much more understandable, don't you agree? It is an exercise that you should be doing whenever writing functions and scripts in your career.

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU
Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. "Introduction à R et au tidyverse." Accessed May 23, 2022. https://juba.github.io/tidyverse
- Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

13

Functions: testing and checking values

13.1 Learning objectives

1. You will learn about using conditional checks to make your functions robust and more reliable.

13.2 Introduction to the dataset

In this lesson, we will continue to to use the COVID-19 dataset containing the results from a serological survey conducted in Yaounde, Cameroon in late 2020.

```
yaounde <- read_csv(here("ch04_functions/data/yaounde_data.csv"))
yaounde
```

13.3 Testing and checking values

As you start to become more confident and write more complex functions, you'll eventually get to the point where you don't remember exactly every detail of how your function works. When this happens, it's easy to call your function with invalid inputs and provide wrong arguments. To avoid this problem, it's often useful to make constraints explicit.

For example, let's recap a shorter version of our`BMI` function.

```
BMI <- function(weight, height) {
  x <- weight / (height ^ 2)
  return(x)
}
```

What happens if the arguments `weight` and `height` are not the same length?

```
u <- yaounde$weight_kg[1:6]

v <- yaounde$height_cm[1:3] / 100
```

```
        BMI(u, v)
```

```
## [1] 33.26214 28.04967 22.83951 24.50895 19.57633 20.06173
```

In this case, because of R's vector recycling rules, we don't get an error. This is specially problematic, since we can get undesired results that will mess all our work.

- Remember that in R, vector recycling rules state that if two vectors are of unequal length, the shorter one will be recycled in order to match the longer vector. In our example, the `u` and the `v` vectors have different lengths, and their sum is computed by recycling values of the shorter vector `v`.
- What this means? It means that the BMI of subjects 4, 5 and 6 were calculated using the repeated heights of subjects 1, 2 and 3!

At a first glance, this could be dismissed as a very obvious error. But, sometimes when dealing with multiple data sets and objects, we can commit errors. In very complex functions (specially if we don't comment them enough), we could be inserting errors during the coding.

Because of this, it's considered good practice to always check important preconditions. If some of these conditions don't apply, we should throw an error (with the function `stop()`,type `?stop` or `help(stop)` in the console to learn more about it).

```
BMI <- function(weight, height) {
    if (length(weight) != length(height)) {
    stop("`weight` and `height` must be the same length", call. = FALSE)
  }
x <- weight / (height ^ 2)
  return(x)
}
```

Notice that checks are done via conditional statement using the `if` function. In our example, our function will return an error if the length of `x` and `y` differ.

```
u <- yaounde$weight_kg[1:6]

v <- yaounde$height_cm[1:3] / 100

BMI(u, v)
```

```
## Error: `weight` and `height` must be the same length
```

You should consider carefully what are the important steps in your function and add some conditional checks of your arguments. However, keep in mind that there's a tradeoff between how much time you spend making your function robust, versus how long you spend writing it. Again, there aren't rules written in stone and you will have to use your judgement on this. For example, take another look at our `BMI` function. Imagine if we also added a `na.rm` argument to is.

```
BMI <- function(weight, height, na.rm = FALSE) {
    if (length(weight) != length(height)) {
    stop("`weight` and `height` must be the same length", call. = FALSE)
  }
  if (!is.numeric(height)) {
    stop("`weight` must be numerical", call. = FALSE)
  }
  if (!is.numeric(height)) {
    stop("`weight` must be numerical", call. = FALSE)
  }
  if (!is.logical(na.rm)) {
    stop("`na.rm` must be logical")
  }
  if (length(na.rm) != 1) {
    stop("`na.rm` must be length 1")
  }

  if (na.rm) {
    miss <- is.na(weight) | is.na(height)
    weight <- weight[!miss]
    height <- height[!miss]
  }

  x <- weight / (height ^ 2)

  return(x)
}
```

Although this is very comprehensive, it is just too much: **it is a lot of extra work for little additional gain**. A useful compromise is the built-in function `stopifnot()`. It checks that each argument is `TRUE`, and produces a generic error message if not.

```
BMI <- function(weight, height, na.rm = FALSE) {
  stopifnot(length(weight) == length(height))
  stopifnot(is.numeric(weight), is.numeric(height))
  stopifnot(is.logical(na.rm), length(na.rm) == 1)

  if (na.rm) {
    miss <- is.na(weight) | is.na(height)
    weight <- weight[!miss]
    height <- height[!miss]
  }

  x <- weight / (height ^ 2)

  return(x)
}
```

WATCH OUT

Note that when using `stopifnot()` you must assert what is supposed be `TRUE` rather than checking for what might be wrong.

PRACTICE

(in RMD)

In the lesson 2, we create an function to categorize the value of an age column. One possible solution is:

```
cat_age <- function(data, age_column, age_range, age_name){

  age_groups <- cut(data %>% pull(age_column),
                    breaks = age_range,
                    labels = age_name, include.lowest = T, right
        = F)

  data['age_group'] = age_groups

  return (data)
}
```

To ensure that the parameters are filled correctly before applying the `cut` function we can add in the function some verification `if` statements as shown above. Look at the four chunks of code below and select the letter referring to the right chunk that you think that should be used in the `cat_age` function to verify if the parameters passed in the function have the exact type and length.

- A

```
stopifnot(is.data.frame(data))
stopifnot(is.numeric(age_column))
```

```
        stopifnot(is.vector(age_range) &&is.vector(age_name))
        if (!age_column %in% colnames(data)) {
  stop(sprintf("The 'data' data frame does not have a column
        called %s.", age_column), call. = FALSE)
  }
if (!length(age_name) == length(age_name)) {
  stop("The length of age_name should be equals to the length of
        age_range.", call. = FALSE)
  }
```

- B

```
stopifnot(is.vector(data))
stopifnot(is.character(age_column))
stopifnot(is.vector(age_range) && is.vector(age_name))

if (!age_column %in% colnames(data)) {
  stop(sprintf("The 'data' data frame does not have a column
        called %s.", age_column), call. = FALSE)
  }

if (!length(age_name) == length(age_name) + 1 ) {
  stop("The length of age_name should be equals to the length of
        age_range more 1.", call. = FALSE)
  }
```

- C

```
stopifnot(is.data.frame(data))
stopifnot(is.numeric(age_column))
stopifnot(is.vector(age_range) && is.vector(age_name))

if (age_column %in% colnames(data)) {
  stop(sprintf("The 'data' data frame does not have a column
        called %s.", age_column), call. = FALSE)
  }

if (!length(age_name) == length(age_name) - 1 ) {
  stop("The length of age_name should be equals to the length of
        age_range minus 1.", call. = FALSE)
  }
```

- D

```
stopifnot(is.data.frame(data))
stopifnot(is.character(age_column))
stopifnot(is.vector(age_range) && is.vector(age_name))

if (!age_column %in% colnames(data)) {
  stop(sprintf("The 'data' data frame does not have a column
        called %s.", age_column), call. = FALSE)
```


```
  }
if (!length(age_name) == length(age_name) - 1 ) {
  stop("The length of age_name should be equals to the length of
       age_range minus 1.", call. = FALSE)
  }
```

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. “Introduction à R et au tidyverse.” Accessed May 23, 2022. https://juba.github.io/tidyverse

Wickham, Hadley; Grolemund, Garrett. “R for Data Science.” Accessed May 25, 2022. https://r4ds.had.co.nz/

- Wickham, Hadley; Grolemund, Garrett. “The tidyverse style guide.” Accessed May 30, 2022. https://style.tidyverse.org/

14

Conditionals: introduction

14.1 Learning objectives

1. You will learn about conditional execution of code using the `if` statement.
2. You will learn more complex conditional execution of code using the `if/else` statement.

14.2 Conditional statement `if`

An `if` statement allows you to conditionally execute code. To define an `if` statement, we need to define a condition, that's simple a comparison of values. When comparing values in R, the output is a **BOOLEAN** value. There are two types of boolean values in R, `TRUE`(or `T`), and `FALSE` (or `F`).

REMINDER

We have the following operators to compare values in R, they are known as **relational operators**:

- == : Equal to.
- != : Not equal to.
- < : Less than. This operator only can be applied with numeric values or dates.
- > : Greater than. This operator only can be applied with numeric values or dates.
- <= : Less than or equal to. This operator only can be applied with numeric values or dates.
- >= : Greater than or equal to. This operator only can be applied with numeric values or dates.

Look at some examples of conditions and your returned values:

- `'example' == 'example'` returns `TRUE`;
- `5 > 6` returns `FALSE`;
- `1 != 1` returns `FALSE`.

The basic structure of an `if` statement looks like:

```
if (condition) {
  # code executed when condition is TRUE
}
```

If your `if` statement is short and simple, you can structure it like this:

```
if (condition) # code executed when condition is TRUE
```

KEY POINT

Note that the code after the **`if`** statement is executed **only** when the condition is **`TRUE`**.

If you need to get help on the `if` you need to surround it in backticks: ``?`if```.

Here's a simple example using the `if` statement. It will only display a message if your first name is `"Laure"`.

```
first_name <- "Laure"

if (first_name == "Laure") {
    message("Hello!")
}
```

or

```
first_name <- "Laure"

if (first_name == "Laure") message("Hello!")
```

Surely, we could use the `if` to create useful tests or functions for our analyzes.

REMINDER

Sometimes we may want to combine different conditions, in this case, we use the following **logical operators**:

- `!`: This operator return the oposite boolean value of a condition. For example, `5>3` returns `TRUE`, while `!5>3` returns `FALSE`.

REMINDER

- `&&`: This operator can be used to combine two expressions. In this case, the returned value is only `TRUE` when the two conditions are `TRUE`. For example, `(5>3) && (7>3)` return `TRUE`, since the two conditions return `TRUE`, while `(5>3)&&(1>3)` returns `FALSE`, since `1>3` is `FALSE`.
- `&`: The `&&` return just one boolean vale. So, if in the condition we have vectors, the boolean value will be decide according to the first values in each vector. But, if we use the `&` operator, it will compare the values element by element. For example, `c(TRUE, FALSE, FALSE) & c(TRUE, TRUE, FALSE)` returns `TRUE FALSE FALSE`, while `c(TRUE, FALSE, FALSE) && c(TRUE, TRUE, FALSE)` returns `TRUE`.
- `||`: This operator can be used to combine two expressions. In this case, the returned value is `TRUE` when one of the conditions are `TRUE`. For example, `(5>3) || (7>3)` return `TRUE`, since the two conditions return `TRUE`, while `(5>3) || (1>3)` returns `TRUE`, since, at least, `5>3` is `TRUE`.
- `|`: The `||` return just one boolean vale. So, if in the condition we have vectors, the boolean value will be decide according to the first values in each vector. But, if we use the `|` operator, it will be compare the values element by element. For example, `c(TRUE, FALSE, FALSE) | c(TRUE, TRUE, FALSE)` returns `TRUE TRUE FALSE`, while `c(TRUE, FALSE, FALSE) || c(TRUE, TRUE, FALSE)` returns `TRUE`.

PRACTICE

(in RMD)

- Let `x = runif(n = 1, min = 36.5, max = 42)` be the measure of the body temperature of a patient that arrived at the emergency of your hospital. Look at the four chunks of code below and select the letter referring to the chunk with an `if` statement that returns the message: "The patient has fever", if the temperature is equal or above 38 degrees Celsius.
- A

```
x = runif(n = 1, min = 36.5, max = 42)

if (x > 38) {
```

```
    message("The patient has fever.")       }
```

- B

```
x = runif(n = 1, min = 36.5, max = 42)

if (x = 38) {
    message("The patient has fever.")
  }
```

- C

```
x = runif(n = 1, min = 36.5, max = 42)

if (x >= 38) {
    message("The patient has fever.")
  }
```

- D

```
x = runif(n = 1, min = 36.5, max = 42)

if (x <= 38) {
    message("The patient has fever.")
  }
```

```
## Correct! You are fantastic!
##   1   2   3
```

```
## Remember that the code after the `if` statement is executed
only when the condition is TRUE
```

14.3 Conditional statement `if/else`

In this special case of the `if` statement, there will be a statement `else` that is only executes if the given condition `if` is `FALSE`. It's basic structure looks like this:

```
if (condition) {
  # code executed when condition is TRUE
} else {
  # code executed when condition is FALSE
}
```

Note that the code after the **`else`** statement is executed **only** when the condition is **`FALSE`**.

At this point, it will be difficult to understand your code if you structure your `if/else` in a single line, so avoid doing it.

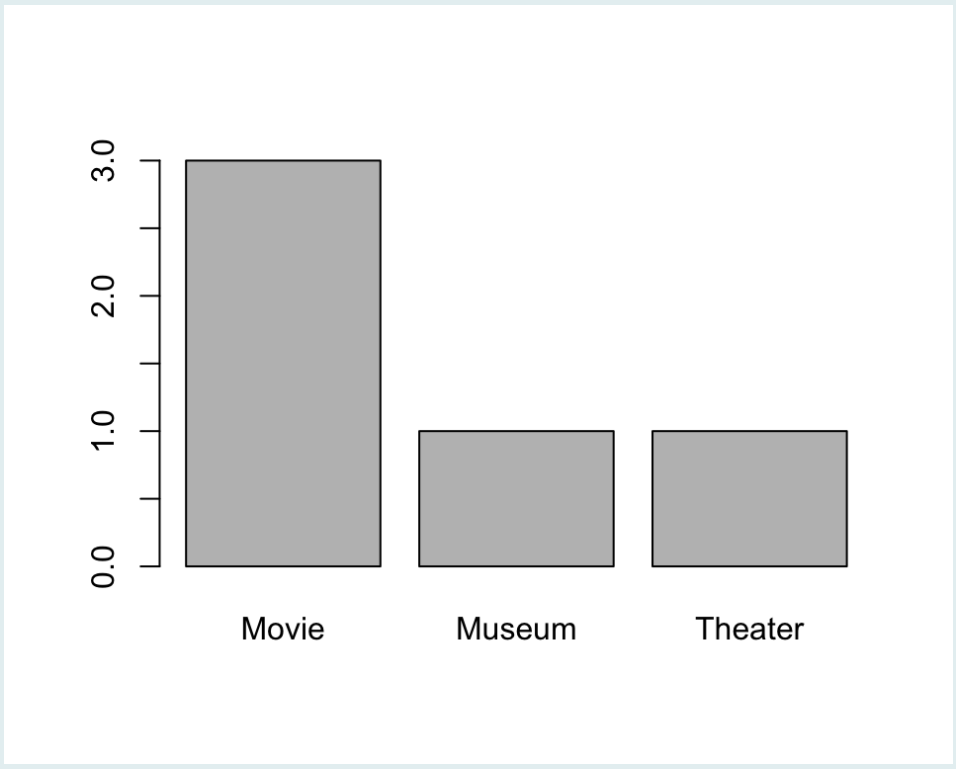
Let's go back to our previous example and make sure to greet anyone that comes by:

```
first_name = 'Claire' # Type the first name here

if (first_name == "Laure") {
    message("Hello!")
  } else {
    message("Oh, hi there you too!")
  }
```

You will find that the `if/else` will be very useful to perform two different actions depending on the value of an argument. For instance, the following code will generate two different graphs depending on the type of vector `x`.

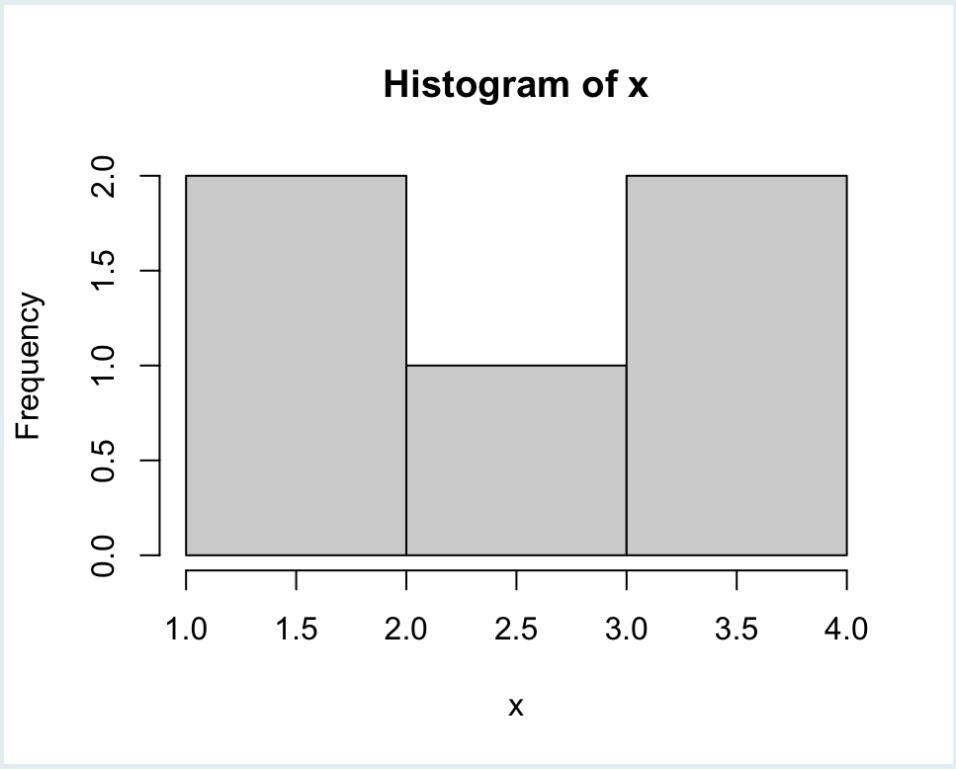
When `x` is a vector of strings is returned a bar plot.

```
x = c("Movie", "Theater", "Movie", "Movie", "Museum")

if (is.character(x)) {
      barplot(table(x))
    } else {
        hist(x)
    }
```



When `x` is a numeric vector, a histogram is returned.

```
x = c(2, 4, 4, 3, 1)

if (is.character(x)) {
      barplot(table(x))
    } else {
        hist(x)
    }
```



- Let `x = runif(n = 1, min = 18, max = 32)` be the BMI of a patient. Look at the four chunks of code below and select the letter referring to the chunk with an `if/else` statement that returns the message: "Your BMI indicates overweight.", if the BMI is above 25, and, otherwise, the message: "Your BMI indicates that you are not overweight".

- A

PRACTICE (in RMD)

```
x = runif(n = 1, min = 18, max = 32)

if (x >= 25){
    message("Your BMI indicates overweight.")
  } else {
    message("Your BMI indicates that you are not overweight")
  }
```

- B

```
x = runif(n = 1, min = 18, max = 32)

if (x < 25){
    message("Your BMI indicates overweight.")
  } else {
```

```
    message("Your BMI indicates that you are not overweight")
  }
```

- C

```
x = runif(n = 1, min = 18, max = 32)

if (x <= 25){
    message("Your BMI indicates overweight.")
  } else {
    message("Your BMI indicates that you are not overweight")
  }
```

- D


```
x = runif(n = 1, min = 18, max = 32)

if (x > 25){
    message("Your BMI indicates overweight.")
  } else {
    message("Your BMI indicates that you are not overweight")
  }
```

```
## Correct! You are excellent!
##   1   2   3
```

```
## Remember that the code after the `if` statement is executed
only when the condition is TRUE,
## and the code after the `else` statement is executed only
when the condition is FALSE.
```

14.4 Conditional statement `if/else if/else`

There is the possibility to use `else if` to add additional conditions. As soon as a condition is `TRUE`, the corresponding block of code is executed. The last block `else` is only executed if none of the conditions are `TRUE`. It's basic structure looks like this:

```
if (this) {
  # do that
  } else if (that) {
```

```
    # do that          } else {          #          }
```

This way, you can chain multiple `if` statements together:

```
x = c(TRUE, TRUE, FALSE, TRUE)

if (is.character(x)) {
  barplot(table(x))
} else if (is.numeric(x)) {
    hist(x)
} else {
   message("The type of `x` does not allow a graph!")
}
```

With this, your imagination is your only limit to what you can do with a function. However, avoid having a very long series of chained `if` statements. If this is the case, you should consider rewriting your chain/function.

One thing to pay attention it that only the block of the first true condition is executed. Therefore, **the order of the conditions is extremely important.** In the following example, the second block is never executed and therefore the second message will never be displayed.

```
x = 50

if (x > 30) {
    message("It is hot outside.")
} else if (x > 40) {
    message("Time to go to the beach!")
} else {
  message("Weather is just fine")
}
```

- Let `x = runif(n = 1, min = 36.5, max = 42)` be the measure of the body temperature of a patient that arrived at the emergency of your hospital. Continuing with our last practice, select the chunk of code with an `if/else if/else` statement that returns a message saying that the patient does not have fever, that the patient has fever if the temperature is equal or above 38 degrees Celsius, and that the patient needs urgent attention if the temperature is equal or above 40 degrees Celsius.
- A

```
x = runif(n = 1, min = 36.5, max = 42)

if (x >= 38) {
  message('The patient needs urgent attention.')
} else if (x >= 40) {
  message('The patient has fever.')
} else {
    message('The patient does not have fever.')
}
```

- B

```
x = runif(n = 1, min = 36.5, max = 42)

if (x >= 40) {
  message('The patient needs urgent attention.')
} else if (x >= 38) {
  message('The patient has fever.')
} else {
    message('The patient does not have fever.')
}
```

- C


```
x = runif(n = 1, min = 36.5, max = 42)

if (x > 40) {
  message('The patient needs urgent attention.')
} else if (x >= 38) {
  message('The patient has fever.')
} else {
    message('The patient does not have fever.')
}
```

- D

```
x = runif(n = 1, min = 36.5, max = 42)

if (x >= 38) {
  message('The patient needs urgent attention.')
} else if (x > 40) {
  message('The patient has fever.')
} else {
    message('The patient does not have fever.')
}
```

```
## Correct! You are great!
##   1   2   3
```


```
## Remember that the order of the code blocks in an `if/else if/else` statement is important!
```

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. “Introduction à R et au tidyverse.” Accessed May 23, 2022. https://juba.github.io/tidyverse

Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

15

Conditionals: advanced

15.1 Learning objectives

1. You will learn more how to code complex chains of code using the `if/else if/else` statement.

2. You will learn the difference between `if/else`, `ifelse()` and `dplyr::if_else()`.

15.2 Conditional statement `if/else`, and the functions `ifelse` and `if_else`

In R, it is possible to use `if/else` or the base function `ifelse` (the `dplyr` package has the equivalent `if_else` function). Although both are very similar, the two are very different though.

- The `if/else` statements are used when you test a single condition and you want to execute different blocks of code depending on its result.

Suppose we have an object `x` that contains a **single** value, and we want to display a different message depending on whether it is less or greater than 30. In this case, we use `if/else`

```
temperature <- 32

if (temperature >= 30) {
    message("Temperature is above 30 degrees Celsius")
} else {
    message("Temperature is below 30 degrees Celsius")
}
```

- The function `ifelse` applies a test to **all** elements of a **vector**. It returns a vector whose elements depend on the result of each test.

```
temperatures <- runif(n = 10, min = 20, max = 42)

temperature_test <- ifelse(test = temperatures >= 30,
                           yes = "Temperature is above 30 degrees Celsius",
                           no = "Temperature is below 30 degrees Celsius")

temperature_test
```

```
##  [1] "Temperature is above 30 degrees Celsius" "Temperature is above 30
degrees Celsius" "Temperature is above 30 degrees Celsius"
##  [4] "Temperature is above 30 degrees Celsius" "Temperature is below 30
degrees Celsius" "Temperature is below 30 degrees Celsius"
##  [7] "Temperature is above 30 degrees Celsius" "Temperature is below 30
degrees Celsius" "Temperature is below 30 degrees Celsius"
## [10] "Temperature is above 30 degrees Celsius"
```

The `if_else` function from the `dplyr` package is faster than the base `ifelse`. It is stricter also, because it checks if the `TRUE` and `FALSE` conditions are of the same type. It works nicely inside a `mutate` function (also, from the `dplyr` package). It also allows you to control missing values.

```
library(dplyr)

temperatures <- runif(n = 10, min = 20, max = 42)

temperatures[4] <- NA

df_temperature <- as.data.frame(temperatures)

df_temperature %>%
  mutate(temperature_test = if_else(condition = temperatures >= 30,
                                    true = "Temperature is above 30 degrees
        Celsius",
                                    false = "Temperature is below 30 degrees
        Celsius",
                                    missing = "Missing value!"))
```

You might be wondering what does "checking if the `TRUE` and `FALSE` conditions are of the same type" means. Check out this example. It returns an error since the `false` argument is not a character vector, but a numeric vector.

```
df_temperature %>%
  mutate(temperature_test = if_else(condition = temperatures >= 30,
                                    true = "Temperature is above 30 degrees
        Celsius",
                                    false = 999,
                                    missing = "Missing value!"))
```

```
## Error in `mutate()`:
## ! Problem while computing `temperature_test = if_else(...)`.
```

```
## Caused by error in `if_else()`:
## ! `false` must be a character vector, not a double vector.
```

If you need to categorize a number into a large number of factors, you might want to use the function `cut()` instead of a chain of `if/else if/else`. Read the help of `?cut` to understand it better. `cut` is a very useful function to categorize a numeric vector, such as a vector of BMI values into the categories established in the literature.

- We passed the `x` vector as an argument of numeric vectors to the function.
- We passed a vector of cut points for the intervals as the argument called `breaks`.
- Finally, the argument `labels` received the names of the categories which will be created by the argument `breaks`.


```
patient_temperatures <- runif(n = 10, min = 36.5, max = 43)

cut(x = patient_temperatures,
    breaks = c(-Inf, 37.5, 38, 40, 42, Inf),
    labels = c("No fever", "Fever", "High fever", "Immediate
        attention", "Allert"))
```

```
##  [1] High fever          Fever               High fever
Allert              High fever          Immediate attention
Allert
##  [8] High fever          No fever            High fever
## Levels: No fever Fever High fever Immediate attention
Allert
```

- Let `x = runif(n = 10, min = 36.5, max = 42)` be a vector with body temperature measures of ten different patients that arrived at the emergency of your hospital. Use the `ifelse` or the `if_else` functions to write a test that identifies if a patient has fever when their temperature is equal or above 38 degrees Celsius. If the temperature is below 38, the patient does not have fever. The output messages should be: 'The patient has fever.', or 'The patient does not have fever.'.

This time, since you are creating a vector of strings, *do not* use the function `message`.

15.3 Advanced stuff regarding `if` conditions

This session draws heavily from the excellent advice from Hadley Wickham and Garrett Grolemund's "R for Data Science"https://r4ds.had.co.nz/ book.

The `condition` in your chain must evaluate to either `TRUE` or `FALSE`. If it's a vector or if it's an `NA`, you'll get an error.

```
x <- c(1, 2, 1)

if (x) {}
```

```
## NULL
```

```
if (NA) {}
```

```
## Error in if (NA) {: missing value where TRUE/FALSE needed
```

Be very careful when using logical operators to test conditions.

You should **never** use `|` (or) or `&` (and) in an if statement. These are vectorised operations that apply to multiple values, and that's why we use them inside the `filter()` function of the `dplyr` package. This means that they can be applied to vectors and will return a vector of `TRUE` and `FALSE`. Remember that `if`, `if/else` and `if/else if/else` are not applied over vectors!

Inside `if`, `if/else` and `if/else if/else` statements, we should use `||` (or) and `&&` (and) as operators to combine multiple logical expressions. As soon as `||` sees the first `TRUE` it returns `TRUE` without computing anything else. As soon as `&&` sees the first `FALSE` it returns `FALSE`. This means that the double operators `||` and `&&` can only return a single value, and if given vectors they will only use the first value of each.

Applying the & operator over a vector will return a vector of TRUE and FALSE values


```
x <- 1:5

x > 0 & x <= 2
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE
```

Applying the && operator over a vector will return just the first logical value (in this case, TRUE) and a warning message


```
x <- 1:5

x > 0 && x <= 2
```

```
## [1] TRUE
```

```
#> Warning messages:
#> 1: In x > 0 && x <= 2 : 'length(x) = 5 > 1' in
        coercion to 'logical(1)'
```

When we pass a test to an `if` statement, it is supposed to return a single value `TRUE` or `FALSE`. A common mistake, especially when you are in a function (more on functions in a later lesson), is to apply an `if` statement to a vector. When this happens, R displays an error.

```
x <- 1:5


if (x >= 5) {
    message("x is >= 5")
}
```

If you do have a logical vector, you can use `any()` or `all()` to collapse it to a single value.

```
x <- c(TRUE, TRUE, FALSE, TRUE, FALSE)
```

Wrong

```
# wrong
if (x == TRUE) message("x is true")
```

Correct

```
# correct
if (any(x) == TRUE) message("x is true") else message("x is
        false")
if (all(x) == TRUE) message("x is true") else message("x is
        false")
```

Also, be careful when checking conditions for equality. The == operator is vectorised, which means that you can get a vector of outputs. You might have noticed that we used the == operator in almost all examples, but now you will understand better when to use it in a conditional statement:

To use the == operator, make sure to:

- Either check the `length` of the object is already 1.
- Collapse the object with `all` or `any`.
- Or use the non-vectorised `identical`.

The function `identical` is very strict: it always returns either a single `TRUE` or a single `FALSE`. It also doesn't coerce types. This means that you need to be careful when comparing integers and doubles:

```
x <- c("Sarah", "Sahra", "Sara")

y <- "Sara"

any(x == y)
```

```
## [1] TRUE
```

```
all(x == y)
```

```
## [1] FALSE
```

```
identical(x, y)
```

```
## [1] FALSE
```

Finally, make sure that when checking conditions involving numbers, you should be wary of floating point numbers. In these cases, you might want to use `dplyr::near` for comparisons.

```
x <- sqrt(2) ^ 2

x
```

```
## [1] 2
```

```
x == 2
```

```
## [1] FALSE
```

Actually, `x` is not 2, as you just saw. It is a very, very small number:

```
x - 2
```

```
## [1] 0.00000000000000004440892
```

In such cases, make sure to remember to use `dplyr::near` (if this is what you really want your condition to be):

```
dplyr::near(x, 2)
```

```
## [1] TRUE
```

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA
Data Scientist at the GRAPH Network and fellowship as Public Health

researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. "Introduction à R et au tidyverse." Accessed May 23, 2022. https://juba.github.io/tidyverse
- Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

16

Loops: introduction

16.1 Learning objectives

1. You will learn how to create an iteration to repeat your code over multiple objects.

16.2 Introduction to the dataset

In this lesson, we will use the data comprise of 136 cases of influenza A H7N9 in China.

```
flu <- read_csv(here("ch05_loop_conditionals/data/fluH7N9_china_2013.csv"))
flu
```

16.3 Repeating instructions in a loop

Loops make it possible to repeat code several times, either according to a condition or according to the elements of a vector.

You will learn in the **functions** lesson that one of the most important tools to improve reproducibility and reduce code duplication is creating your own functions. Another important tool that will enable you to reduce code duplication is iteration over a loop. This means that the days of copy-and-paste the same code to repeat the same operation on different columns, or even in different data sets, are over!

16.4 `for` loop

The `for` loop is defined by the statement `for`. The principle behind it is that it will iterate a code over all the items in a vector.

For example, given a vector defined as `c(1,2,3,4,5)`, you can iterate over each value of this vector and print each value using a `for` statement:

```
v = c(1, 2, 3, 4,5)
for (i in v) {              # 1. sequence

  print(i)                  # 2. body

}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

The simplest `for` loop will have at least two elements. Here we will refer to these elements as `sequence`, and `body`. In the `sequence`, we will define a vector/sequence in which the values will be iterated, and in the `body` we will define operations that it will be done with each of the iterated values.

The `for` statement can iterate over any value in a vector, numeric or not. For example, instead of use `c(1,2,3,4,5)`, we could use: `c('Ana', 'Benedict', 'Carlos', 'Daniel', 'Eduardo')` and iterate over a list of names:

```
v = c('Ana', 'Benedict', 'Carlos', 'Daniel', 'Eduardo')
for (i in v) {              # 1. sequence

  print(i)                  # 2. body

}
```

```
## [1] "Ana"
## [1] "Benedict"
## [1] "Carlos"
## [1] "Daniel"
## [1] "Eduardo"
```

You also can iterate over a sequence of numbers as `1:5`:

```
for (i in 1:5) {              # 1. sequence

  print(i)                    # 2. body

}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
## [1] 4
## [1] 5
```

In the examples above, we just iterated over the vector and printed each value. Still, usually, we want to save this result to use in other operations, such as creating a plot or a data frame. In this case, we need to define a vector where we will save the result after each operation.

```
x = c()                    # create an empty vector

for (i in 1:5){            # sequence

  x_i = i^3                # body

  x = append(x, x_i)       # save the result in a vector

}

x
```

```
## [1]   1   8  27  64 125
```

WATCH OUT

You must create your empty element to save the results **outside** the loop. Otherwise, you will save the result of the last iteration.

KEY POINT

That's the usual way that we build and use a `for` loop. We can summarize it in four steps:

- Create an empty element to save the values obtained in the loop;
- Define a sequence that will be iterated;
- Define the operations that will be done inside the loop (the `body`);
- Save the result in the element **created outside** the loop.

Observe that we have a lot of possibilities when using the `for` statements. Let's now see a practice example that can be useful when analysing data.

Imagine we have the following data set to analyse:

```
set.seed(1)
df <- tibble(
  height = runif(n = 20, min = 1.50, max = 2.10),
  weight = runif(n = 20, min = 50, max = 150),
  age = runif(n = 20, min = 18, max = 80),
```

```
  heart_rate = runif(n = 20, min = 40, max = 120),
  bmi = weight / (height ^2)       )                df
```

Now, imagine that you need to compute the median of each variable. Normally, we would copy and paste:

```
median(df$height)
```

```
## [1] 1.86059
```

```
median(df$weight)
```

```
## [1] 94.66773
```

```
median(df$age)
```

```
## [1] 50.49897
```

```
median(df$heart_rate)
```

```
## [1] 78.19586
```

```
median(df$bmi)
```

```
## [1] 31.53887
```

This is a lot of copy-and-paste for just one descriptive statistic. Imagine doing this for the mean, standard deviation, minimum and maximum values?

The `for` loop can easily iterate for all columns!

To do this, we can simply define an output vector for each desired metric, where the result will be saved after each iteration.

```
out_median = c()                                    # create an empty vector

for (column in colnames(df)){                       # sequence

  col_median = median(df[[column]])                 # body

  out_median = append(out_median, col_median)       # save the result
}
```

```
        out_median
```

```
## [1]  1.86059 94.66773 50.49897 78.19586 31.53887
```

Note that you can use any name for the iterator value. In the example above, we use the name `column`.

- Using the `df2` data set below, create a loop that calculates the mean of each variable and stores it in a vector called `df2_means`.Your answer must be the vector `df_means`.


```
set.seed(1)
df2 <- tibble(
  a = rnorm(10),
  b = rnorm(10),
  c = rpois(10, 10),
  d = rnorm(10),
  e = rpois(10, 3),
  f = rnorm(10),
)

df2
```

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. "Introduction à R et au tidyverse." Accessed May 23, 2022. https://juba.github.io/tidyverse
- Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

17

Loops: automating tasks

17.1 Learning objectives

1. You will learn how to use `for` loops to automate tasks in R.

17.2 Introduction to the dataset

In this lesson, we will use some data from the Gapminder project.

```
gapminder = read_csv('https://raw.githubusercontent.com/OHI-Science/data-
        science-training/master/data/gapminder.csv')
gapminder
```

17.3 Automation with `for` loops

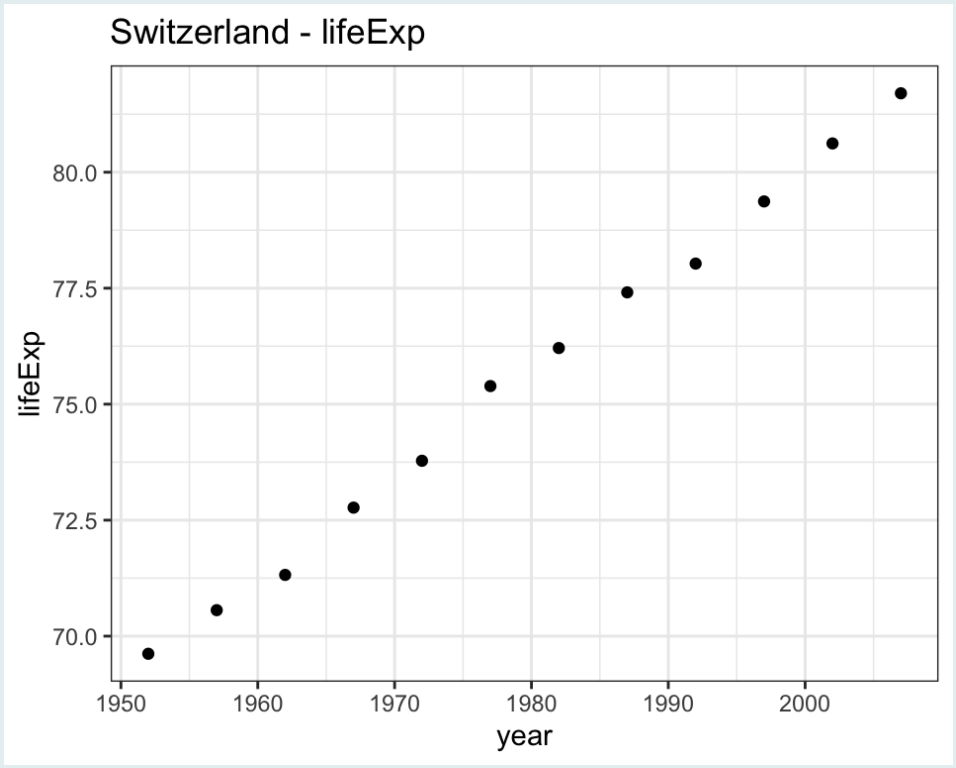
For loops can be really useful to automate tasks in R.

Suppose we want to plot the variation in time of the life expectancy in Switzerland. We can do that using the code below. In the plot, we use as title: `Switzerland - lifeExp` and save the plot in the `images` folder.

```
## filter the country to plot
gap_to_plot <- gapminder %>%
  filter(country == "Switzerland")

## plot
my_plot <- ggplot(data = gap_to_plot, aes(x = year, y = lifeExp)) +
  geom_point() +
  ## add title and save
  labs(title = "Switzerland - lifeExp")

ggsave(filename = 'Switzerland_lifeExp.png', path =
        here("ch05_loop_conditionals/images"), plot = my_plot)
```

```
## Error in grDevices::dev.off(): QuartzBitmap_Output - unable to open file
'/Users/kendavidn/Dropbox/graph_courses/course_projects/further-data-
analysis-with-r-
staging/ch05_loop_conditionals/images/Switzerland_lifeExp.png'
```

```
my_plot
```



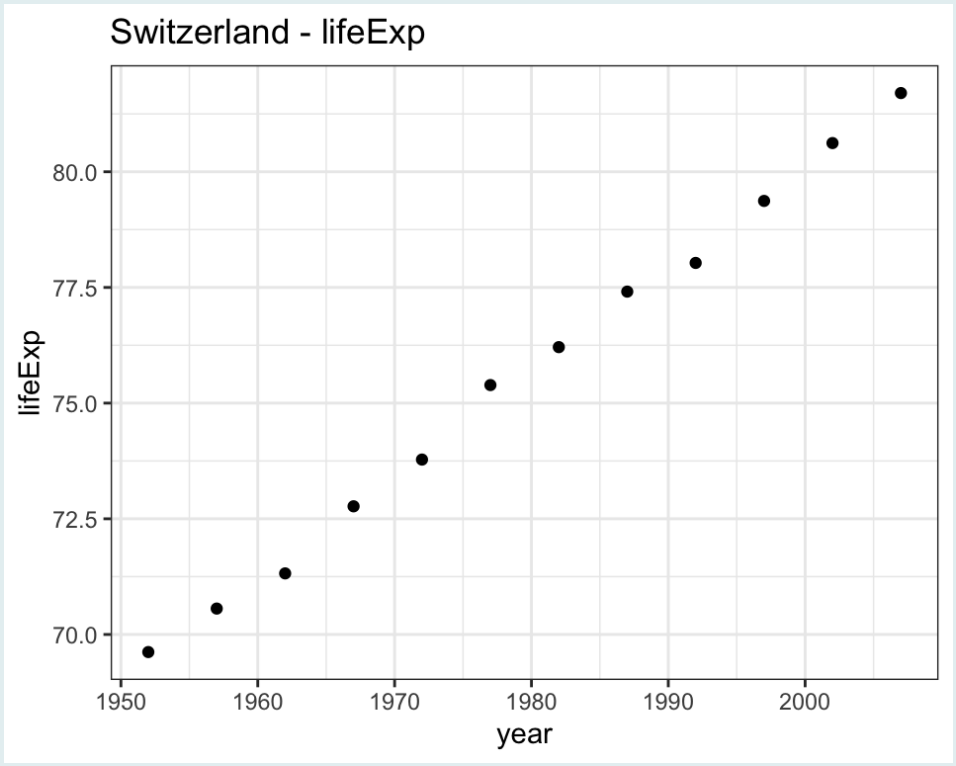
To be able to create this plot with another country, let's replace the 'Switzerland' string by a variable called `ctry`.

```
ctry = 'Switzerland'

gap_to_plot <- gapminder %>%
  filter(country == ctry)

## plot
my_plot <- ggplot(data = gap_to_plot, aes(x = year, y = lifeExp)) +
  geom_point() +
  ## add title and save
  labs(title = paste(ctry, " - ", "lifeExp", sep = ""))

ggsave(filename = paste(ctry, '_lifeExp.png', sep = ""), path =
        here("ch05_loop_conditionals/images"), plot = my_plot)
```

```
## Error in grDevices::dev.off(): QuartzBitmap_Output - unable to open file
'/Users/kendavidn/Dropbox/graph_courses/course_projects/further-data-
```

```
analysis-with-r-
staging/ch05_loop_conditionals/images/Switzerland_lifeExp.png'
```

```
my_plot
```



Suppose now, that we want to create this plot also for 'Brazil', 'Colombia' and 'Angola'. Instead of running the code above three times, we can create a loop that given a vector, change the value of `ctry` after each iteration. Let's do this in the code below:

```
countries = c("Brazil", "Colombia", "Angola")

for (ctry in countries){
  gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes(x = year, y = lifeExp)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", "lifeExp", sep = ""))

  ggsave(filename = paste(ctry, '_lifeExp.png', sep = ""), path =
         here("ch05_loop_conditionals/images"), plot = my_plot)
  }
```

```
## Error in grDevices::dev.off(): QuartzBitmap_Output - unable to open file
'/Users/kendavidn/Dropbox/graph_courses/course_projects/further-data-
analysis-with-r-staging/ch05_loop_conditionals/images/Brazil_lifeExp.png'
```

Based on the code above, select between the chunks of code below, which allows you to generate the plot for a list of countries and columns. The code must allow you to plot any column of the gapminder data set.

- A


```
countries = c("Brazil", "Colombia", "Angola")
cols_plot = c("pop", "lifeExp","gdpPercap")

for (col in countries){
  for (ctry in cols_plot) {

    gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes_string(x ="year", y
        = col)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", col, sep = ""))

  ggsave(filename = paste(ctry, '_' , col, '.png', sep = ""),
        path = here("ch05_loop_conditionals/images"), plot =
        my_plot)
  }
}
```

- B

```
countries = c("Brazil", "Colombia", "Angola")
cols_plot = c("pop", "lifeExp","gdpPercap")

for (ctry in countries){
  for (col in cols_plot) {

    gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes_string(x ="year", y
        = lifeExp)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", col, sep = ""))

  ggsave(filename = paste(ctry, '_' , 'lifeExp', '.png', sep =
        ""), path = here("ch05_loop_conditionals/images"), plot
        = my_plot)
  }
```

```
}
```

- C

```
countries = c("Brazil", "Colombia", "Angola")
cols_plot = c("pop", "lifeExp","gdpPercap")

for (ctry in countries){
  for (col in cols_plot) {

    gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes_string(x ="year", y
        = col)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", "lifeExp", sep = ""))

  ggsave(filename = paste(ctry, '_' , col, '.png', sep = ""),
        path = here("ch05_loop_conditionals/images"), plot =
        my_plot)
  }
}
```

- D

```
countries = c("Brazil", "Colombia", "Angola")
cols_plot = c("pop", "lifeExp","gdpPercap")

for (ctry in countries){
  for (col in cols_plot) {

    gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes_string(x ="year", y
        = col)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", col, sep = ""))

  ggsave(filename = paste(ctry, '_' , col, '.png', sep = ""),
        path = here("ch05_loop_conditionals/images"), plot =
        my_plot)
  }
}
```

```
## Correct! You are laudable!
##   1
```

We can also save the data filtered by country in a `.csv` file by adding the function `write.csv` in the loop.

```
countries = c("Brazil", "Colombia", "Angola")

for (ctry in countries){
  gap_to_plot <- gapminder %>%
    filter(country == ctry)

  ## plot
  my_plot <- ggplot(data = gap_to_plot, aes(x = year, y = lifeExp)) +
    geom_point() +
    ## add title and save
    labs(title = paste(ctry, " - ", "lifeExp", sep = ""))

  setwd(here("ch05_loop_conditionals/data"))

  write.csv(gap_to_plot, paste("gapminder_",ctry, ".csv", sep = ""), row.names
        = FALSE)

  ggsave(filename = paste(ctry, '_lifeExp.png', sep = ""), path =
        here("ch05_loop_conditionals/images"), plot = my_plot)
}
```

```
## Error in grDevices::dev.off(): QuartzBitmap_Output - unable to open file
'/Users/kendavidn/Dropbox/graph_courses/course_projects/further-data-
analysis-with-r-staging/ch05_loop_conditionals/images/Brazil_lifeExp.png'
```

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- The Ocean Health Index Team. "Introduction to Open Data Science." Accessed Jun 10, 2022. http://ohi-science.org/data-science-training/programming.html#automation-with-for-loops

18

Loops: best practices

18.1 Learning objectives

1. You will learn good practices to building a loop.
2. You will learn how to use multiple `for` loops.

18.2 Introduction to the dataset

In this lesson, we will use the data comprise of 136 cases of influenza A H7N9 in China.

```
flu <- read_csv(here("ch05_loop_conditionals/data/fluH7N9_china_2013.csv"))
flu
```

18.3 Good practices building a loop

Let's remember the example of the last lesson:

Imagine we have the following data set to analyse and you need to compute the median of each variable.

```
set.seed(1)
df <- tibble(
  height = runif(n = 20, min = 1.50, max = 2.10),
  weight = runif(n = 20, min = 50, max = 150),
  age = runif(n = 20, min = 18, max = 80),
  heart_rate = runif(n = 20, min = 40, max = 120),
  bmi = weight / (height ^2)
)

df
```

To do this, we can simply define an output vector for each desired metric, where the result will be saved after each iteration.

```
out_median = c()                                    # create an empty vector

for (column in colnames(df)){                       # sequence

  col_median = median(df[[column]])                 # body

  out_median = append(out_median, col_median)       # save the result
}

out_median
```

```
## [1]  1.86059 94.66773 50.49897 78.19586 31.53887
```

In the last example, we defined our output vector as an empty vector using `c()`. However, this is not the optimal way to do that. The optimal way is to define a vector of the exact amount of space necessary and the right type of values that will be stored in the vector.

We can do that using the `vector()` function. This function has two arguments:

- `mode` that must be filled with the type of the vector (""logical", "integer", "double", "character", etc);
- `length`: that must be filled with the length of the vector that we want to create.

To use `c()` or `vector()` results in minimal difference for small loops (as the example above). **But it can save a valuable amount of time when using a bigger loop.**

In the example above, we could define our output using the `vector()` function as:

```
out_median = vector("double", ncol(df))
out_median
```

```
## [1] 0 0 0 0 0
```

When we define a vector using the `vector()`, we will not use the `append()` method over the loop. You can see above that the vector already has the exact length to be used in the loop. So, if you use the `append()` method, you will change the length of the vector.

```
append(out_median, 1)
```

```
## [1] 0 0 0 0 0 1
```

Therefore, when using the `vector()` we need to change the value of the vector by your index. For example, to change the first value of the vector `out_median` we use:

```
out_median = vector("double", ncol(df))

out_median[[1]] = 4

out_median
```

```
## [1] 4 0 0 0 0
```

You might have noticed that we used `[[]]` instead of `[]`. We follow Hadley Wickham's advice that it's better to use `[[]]` even for atomic vectors because it makes it clear that we want to work with a single element.

Now, let's change the definition of the vector in our last example. In the example, we were iterating over the columns names and appending the values in an empty vector. But, now, how we need to change the elements of our vector by index, we will need to iterate over the values of the index. To do this, we will need to change the definition of our sequence. Now we will use `for (i in 1:ncol(df))`.

So, rewriting, we have:

```
out_median = vector("double", ncol(df))                                    #
        create an empty vector

for (i in 1:ncol(df)){                        # sequence

  col_median = median(df[[i]])                  # body

  out_median[[i]] = col_median      # save the result
}

out_median
```

```
## [1]  1.86059 94.66773 50.49897 78.19586 31.53887
```

Note that we have the same output obtained using the `c()`.

In the loop code above, we define the sequence using `1:ncol(df)`, where `ncol(df)` represents the number of columns in `df`. There is an optimal way to do it in R. We can use the `seq_along(x)` function, which create a sequence based on the length of the `x` element in the function.

The `seq_along(x)` is a safe version of the familiar `1:length(x)`. The crucial thing that `seq_along()` does is that if you have a zero-length vector, it will do the right thing:


```
x <- vector("double", 0)

# right
seq_along(x)
```

```
## integer(0)
```

```
# wrong
1:length(x)
```

```
## [1] 1 0
```

Since we warned that it is important to **not** iterate over a zero-length vector deliberately, it's easy to create them accidentally. If you use `1:length(x)` instead of `seq_along(x)`, you're likely to get a confusing error message. The `seq_along(x)` function will prevent this.

So, in our last example, we will change `1:ncol(df)` to `seq_along(df)`.

```
seq_along(df)
```

```
## [1] 1 2 3 4 5
```

So, in each loop iteration, `i` will assume one of the values in `seq_along(df)` (e.g., 1 in the first iteration, 2 in the second iteration, and so on up to i = 5 in the fifth iteration).

Thus, more optimally, our last example can be rewritten as:

```
out_median = vector("double", ncol(df))                                    #
        create an empty vector

for (i in seq_along(df)) {                          # sequence
```

```
          col_median = median(df[[i]])                # body
  out_median[[i]] = col_median     # save the result       }
out_median
```

```
## [1]  1.86059 94.66773 50.49897 78.19586 31.53887
```

and, as expected the result is the same.

To build a `for` loop, follow the steps below:

- Create an empty element to save the values obtained in the loop. Remember to define it **outside** the loop, and that's recommended to use `vector()` instead of `c()`;
- Define a sequence that will be iterated. Remember that's recommended to use `seq_along(x)` instead of `1:length(x)`;
- Define the operations that will be done inside the loop (the `body`);
- Save your result in the element created outside the loop. Remember that's recommended to use `[[]]` instead of `[]` when accessing values by index.

- Using the `flu` data set determine the type of the column `case_id`.
- Using a `for` loop, compute the type of each column in the `flu` data set and store it in a vector. Your answer must be this vector.

We also can concatenate multiple `for` loops. Let's see an example with the `flu` data set.

The `flu` data set represents the cases of Influenza in 2013, in some provinces in China. Using that data set we can create a new data frame with three columns:

- `province`: Representing the province of China where the cases were notified;
- `cases_m`: Representing the cumulative number of cases between the men in this period;
- `cases_f`: Representing the cumulative number of cases between the women in this period.

Let's create this data frame for the provinces 'Anhui', 'Beijing', and 'Shangai'. In order to do this, we need, for each province, filter the data by the province, count the number of `m` and `f` values in the `gender` variable and save it in our data frame.

First, let's use the `group_by` function to aggregate our data set by the values in `province` and `gender` to get the number of cases between men and women in each province.

```
flu_group = filter(flu %>% group_by(province, gender) %>%
                   summarise(obs = n(),
                             .groups = 'drop'))
flu_group
```

Now, let's create a new data frame where we will save the results of our loop. Let's define two vectors, one with the province values (called `prov` in the code) and another with the gender values (called `sex` in the code).

```
# vector with the provinces
prov = c("Anhui", "Beijing", "Shanghai")

# vector with the gender values
sex = c("m", "f")

df_group <- tibble(
  province = prov,
  cases_m = vector("integer", length(prov)),
  cases_f = vector("integer", length(prov)))

df_group
```

We can access the values of a data frame, and replace them, by the number of the row and column that we want to access. For example, we can get the value in the first row (province = 'Anhui') and second column (column `cases_m`), and change it to `10`, using the notation:

```
df_group[[1,2]] = 10

df_group
```

We will use the information in the box above to finish our `for` loop.

```
for (i in seq_along(prov)){

  for (j in seq_along(sex)){
    df_group[[i, j+1]] = filter(flu_group, (province == prov[[i]] & gender ==
        sex[[j]])) $ obs
    }
  }

df_group
```

Notice that we used `j + 1` to access the column of the data frame, because, when `j = 1`, then `sex[[1]] = 'm'`, and the column `cases_m` is the second (2) column of the data frame. In the same way, when `j = 2`, then `sex[[2]] = 'f'`, and the column `cases_f` is the third (3) column of the data frame.

When we have two for `for` loops, first, we will have the first iteration of the first loop and so, we will iterate over all the values in the second loop before iterate the second value of the first for loop.

Contributors

The following team members contributed to this lesson:

DANIEL CAMARA

Data Scientist at the GRAPH Network and fellowship as Public Health researcher at Fiocruz, Brazil
Passionate about lots of things, especially when it involves people leading lives with more equality and freedom

EDUARDO ARAUJO

Student at Universidade Tecnologica Federal do Parana
This is a temporary tagline

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about education.

References

Some material in this lesson was adapted from the following sources:

- Barnier, Julien. "Introduction à R et au tidyverse." Accessed May 23, 2022. https://juba.github.io/tidyverse
- Wickham, Hadley; Grolemund, Garrett. "R for Data Science." Accessed May 25, 2022. https://r4ds.had.co.nz/

19

Geospatial analysis: thematic maps

19.1 Learning objectives

1. Identify two types of **Thematic maps** (choropleth and dot maps) used by epidemiologist to visualize Geospatial data.

2. Create Thematic maps using `{ggplot2}` and the **`geom_sf()`** function.

3. Relate each Thematic map with a **Spatial data** type.

19.2 Prerequisites

This lesson requires the following packages:

```
if(!require('pacman')) install.packages('pacman')
pacman::p_load_gh("afrimapr/afrilearndata")
pacman::p_load(ggspatial,
               ggplot2,
               mdsr,
               terra,
               spData)
```

This lesson requires familiarity with `{ggplot2}`: if you need to brush up, have a look at our introductory course on data visualization.

19.3 Introduction

Spatial aspects of your data can provide a lot of insights into the situation of a certain disease or an outbreak, and to answer questions such as:

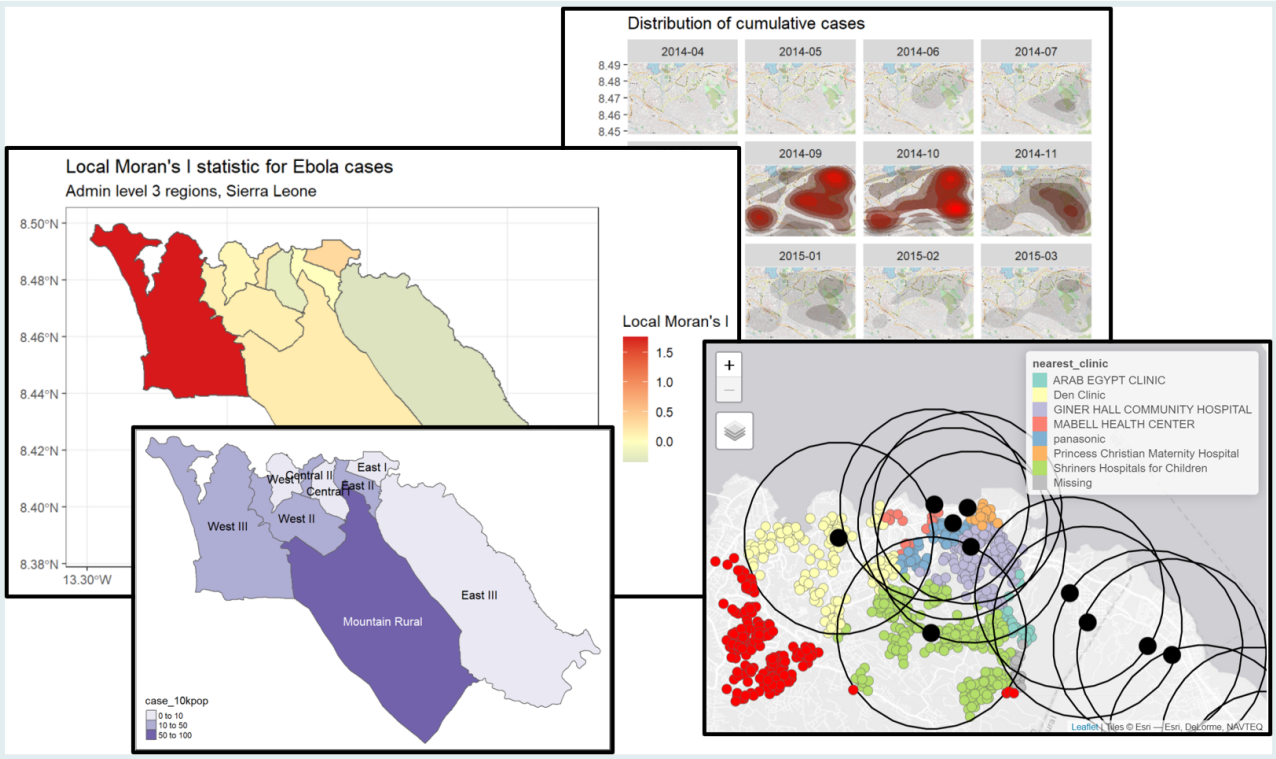
- **Where** are the current disease hotspots?
- How have the hotspots **changed over time**?
- How is the **access** to health facilities? Are any improvements needed?

A Geographic Information System (GIS) is a framework or environment for gathering, managing, analyzing, and visualizing spatial data.

Here we are using the R as a GIS environment to address all these tasks, in order to address the needs of applied epidemiologists in outbreak response.

Thematic maps

Thematic maps portrait geographic patterns about a particular subject theme in a geographic area. In the context of epidemiological outbreaks, these maps can be called *Epidemic maps*.



The most iconic types of thematic maps to visualize your spatial data are *Choropleth maps* and *Dot maps*.

19.4 Choropleth map

What is it?

A *Choropleth map* is a type of thematic map where *colors, shading, or patterns* are used to represent **geographic regions** in relation to the value of an attribute.

For instance a *larger value* could be indicated by a darker color, while a *smaller value* could be indicated by a lighter color.

How to plot it?

Geospatial data can be plotted with the `{ggplot2}` package, using the **`geom_sf()`** function. Information such as *colors* and *shapes* can be depicted using the `aes()` function with the `fill`, `color` and `shape` arguments.

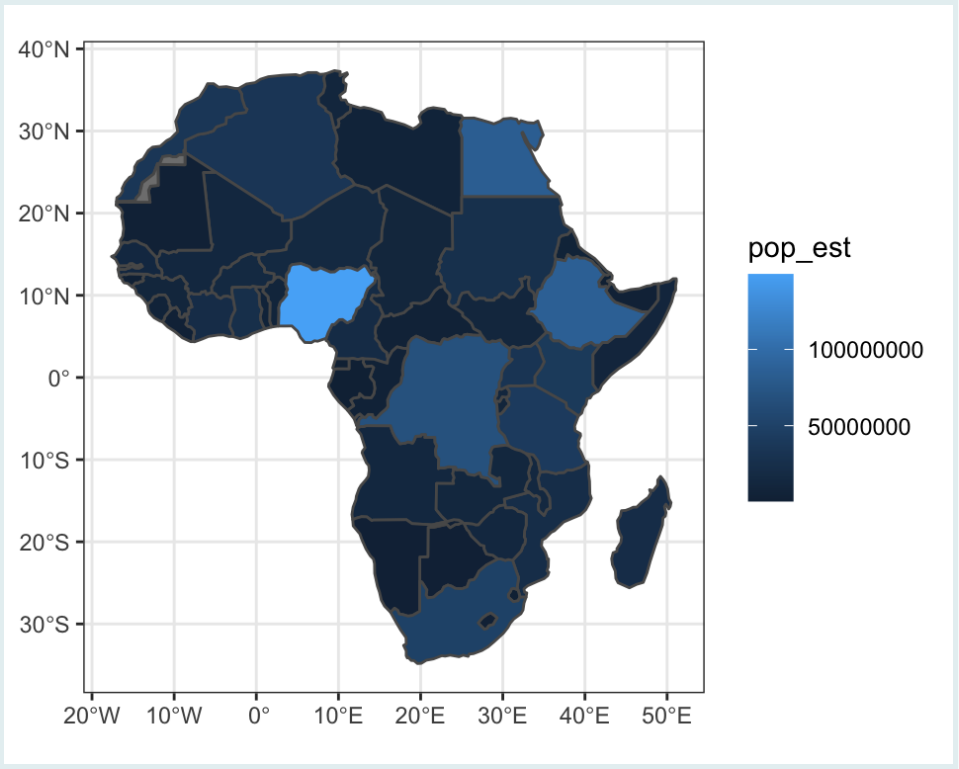
SIDE NOTE

`sf` stands for **"simple features"**, an open standard used to represent a wide range of *geometric shapes*.

A *Choropleth map* will usually require using the `fill` argument. Let's create a Choropleth map!

1. Install the `{afrilearndata}` package
2. Use the `africountries` dataset. It contains the administrative boundaries of all the countries in the African continent.
3. Then, use `{ggplot2}` and the `geom_sf()` function to plot African countries,
4. and `fill` each of them in relation to the estimated population (`pop_est`) of each country:

```
pacman::p_load_gh("afrimapr/afrilearndata")
pacman::p_load(ggplot2)

ggplot(data = africountries) +
  geom_sf(mapping = aes(fill = pop_est))
```



PRACTICE

(in RMD)

Create a Choropleth map with the `world` data from the `{spData}` package to portrait its countries and `fill` them in relation to its population with the `pop` variable.

How to use it?

This type of map is particularly useful when visualizing a variable and how it changes across defined regions or geopolitical areas.

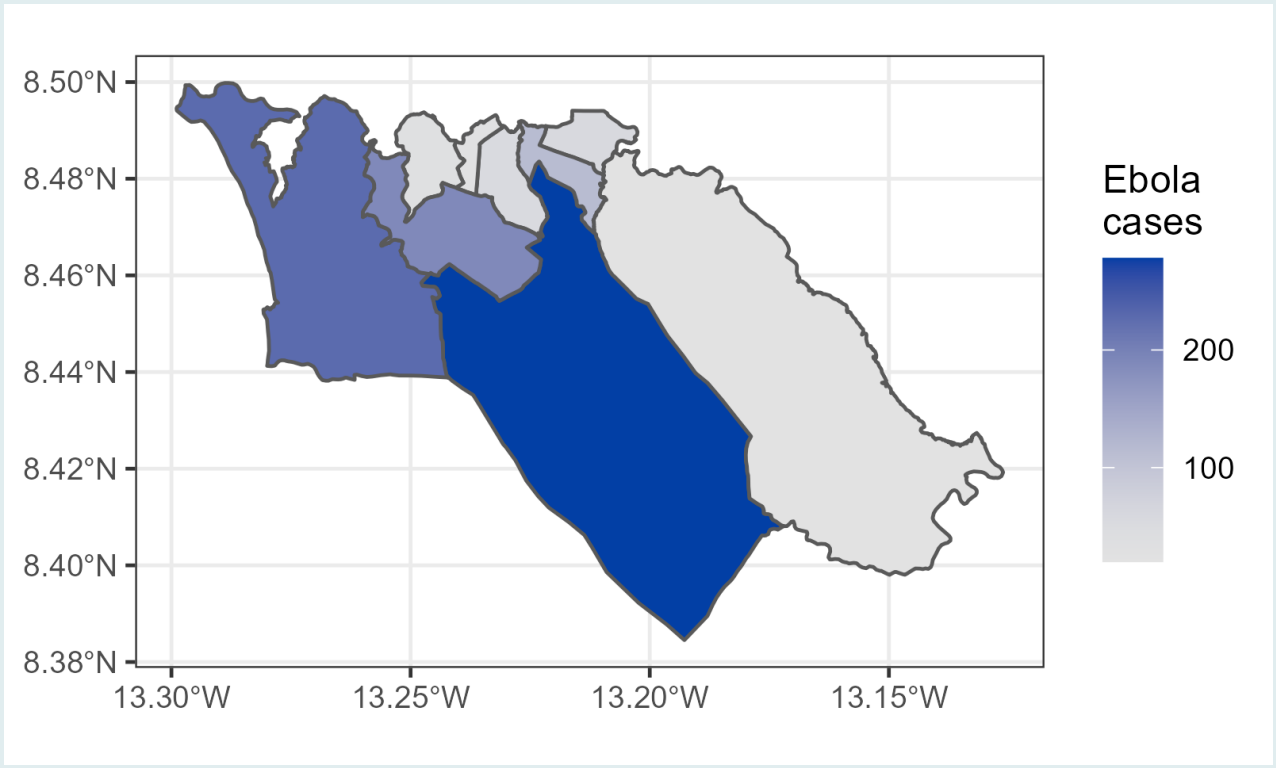


Figure 1. Choropleth map with the number of cases in a simulated Ebola epidemic in Sierra Leona.

In Figure 1, the region of interest (Sierra Leona) has been partitioned into a finite number of subregions (districts) at which the number of cases have been aggregated.

The type of data that *Choropleth maps* visualize is called **Areal data**. This is data that pertains to an enclosed region partitioned into a finite number of areal units with well-defined boundaries. For example, attributes collected by ZIP code, census tract, or the administrative boundary levels of a country (Figure 1).

19.5 Dot map

What is it?

A *Dot map* is a thematic map type that uses **dots** to represent attribute values in your data.

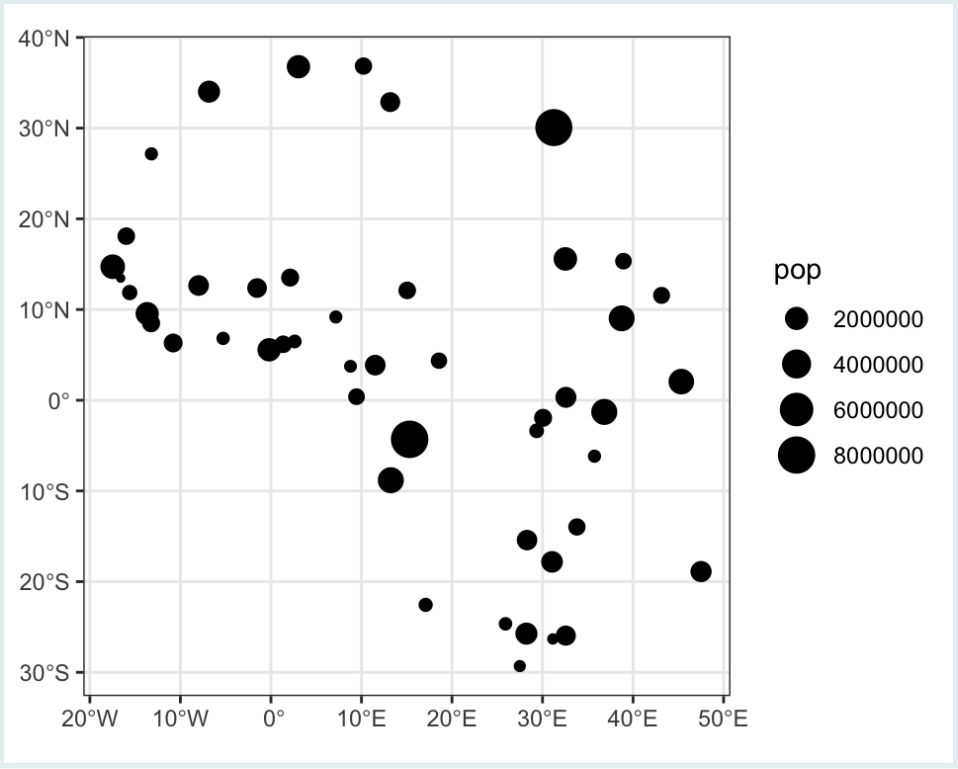
How to plot it?

The *Dot map* could use the `size` or `color` argument. Let's create a Dot map!

1. Use the `africapitals` dataset, also from the `{afrilearndata}` package, which contains the location of capital cities in the African continent.

2. Then, use again `{ggplot2}` and `geom_sf()` to plot these locations,
3. and `size` each of them in relation to their number of inhabitants:

```
ggplot(data = africapitals) +
  geom_sf(mapping = aes(size = pop))
```



PRACTICE

(in RMD)

Create a Thematic map with the `afriairports` object to portrait all its airport locations and `color` them in relation to the `type` variable.

WATCH OUT

Before using `geom_sf()` to create *Thematic maps*, you must verify that your *Spatial data* is an `"sf"` R object using the **`class()`** function:

```
class(africountries)
```

```
## [1] "sf"         "data.frame"
```

In the following lessons, we will learn how to get more of them and even convert foreign objects to `sf`!

How to use it?

This type of map is best used to visualize the *scatter of your data* and visually *scan for clusters*.

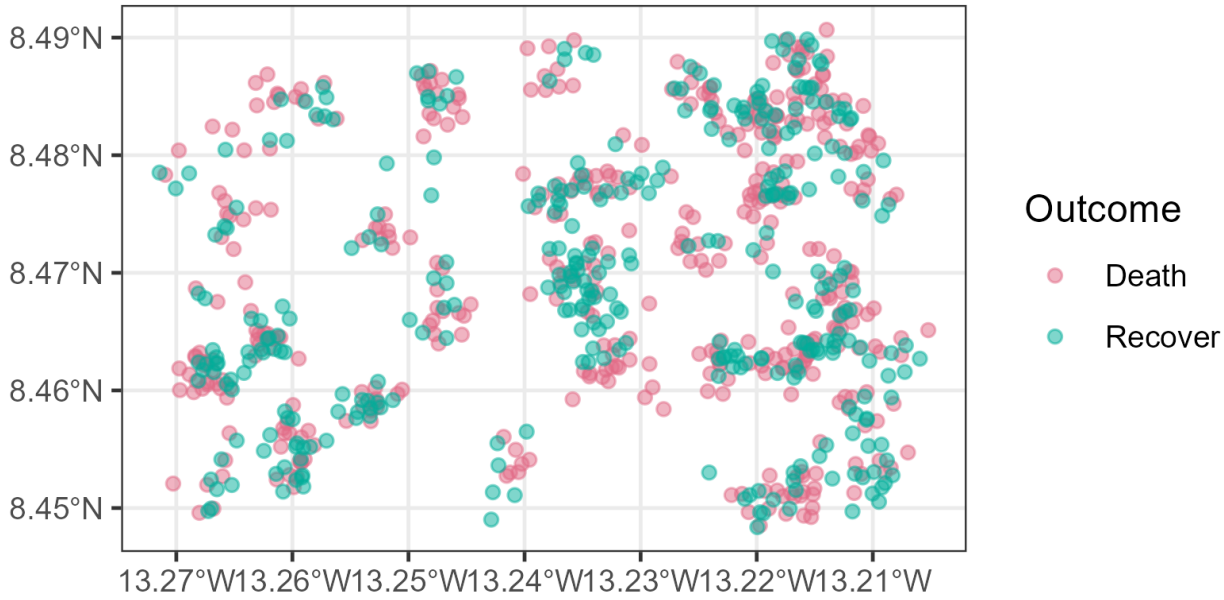


Figure 2. Dot map. Location of simulated Ebola cases in Sierra Leona, colored by each case outcome.

The type of data that *Dot maps* visualize is called **Point patterns**. This is data that register the locations of random events. For example, collecting geographical coordinates of individuals with a given diagnosis (Figure 2): the Ebola cases in Sierra Leone.

SIDE NOTE

Are you bothered by the fact of having just dots and no country lines? That's good ! We will see how to add those using the **spatial data** of **Physical features** very soon.

RECAP

Thematic maps visualize specific *Spatial data* types:

- Choropleth maps visualize *Areal data*.
- Dot maps visualize *Point patterns*.

RECAP

These types refer to the *data generating process* of their spatial information.

Which of the following options of *Thematic map* types:

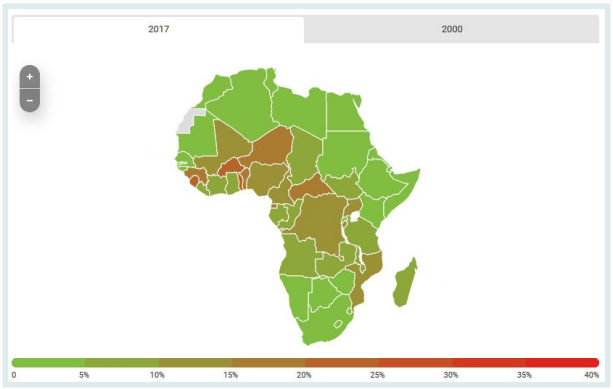
a. `"choropleth_map"`
b. `"dot_distribution_map"`

...corresponds to each of these *Epidemic map* figures?

Your answer should be either "choropleth_map" or "dot_distribution_map".

PRACTICE
(in RMD)

Malaria cases in Africa:



COVID-19 cases in the world:

19.6 Physical features

What are they?

We can complement Thematic maps with spatial *Physical features* like roads, buildings and rivers.

How to plot them?

As an example, we will complement a Choropleth map with the population of African countries, from the `africountries`, with:

- the African trans-continental highway network lines, available in the `afrihighway` dataset from the same package.

```
ggplot() +
  geom_sf(data = africountries, mapping = aes(fill = pop_est)) +
  geom_sf(data = afrihighway)
```

Here, the physical feature `afrihighway` is **above** all the other layers.

But it can also be **below**. For example, we can complement a Dot map with the population in the capital cities of Africa, from the `africapitals` dataset, with the same layer:

```
ggplot() +
  geom_sf(data = afrihighway) +
  geom_sf(data = africapitals, mapping = aes(size = pop, color = pop))
```

This is how you plot another map layer on top of another map.

RECAP

`{ggplot2}` allows to **overlap multiple layers (of maps)** from different data sources to complement Thematic maps.

For this, instead of a *global* specification of data, you need to use a **local** one:

```
# instead of:
ggplot(data = data_global) +
  geom_sf()

# we use:
ggplot() +
  geom_sf(data = data_local_layer_1) +
  geom_sf(data = data_local_layer_2)
```

The *order* of the layers (below or above) will depend on the aim of the plot.

Create a map with the `world` dataset. Then, overlap it with the African trans-continental highway network lines from the `afrihighway` dataset.

How to use it?

We can also add *trajectory data* like records of the location of moving objects at various times. Both of them are depicted in a map using **Lines**.

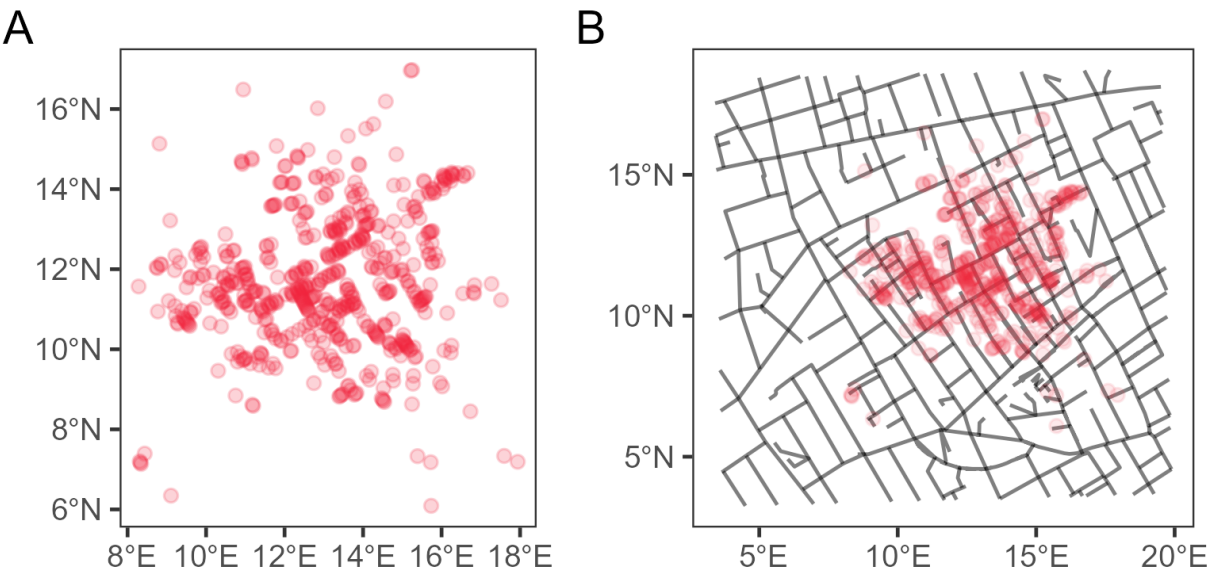


Figure 3. (A) John Snow's Dot map. (B) John Snow's Dot map complemented with the city street roads.

In this example, we replicate John Snow's *Dot map* with the locations of deaths of the 1854 London cholera outbreak. We complemented this map with a *physical feature* like the *street roads* of the city: subfigure B is much more readable than subfigure A right?!

In Figure 3B, the *physical feature* is used as background, below the Dot map. This is why it looks much more readable.

Basemaps

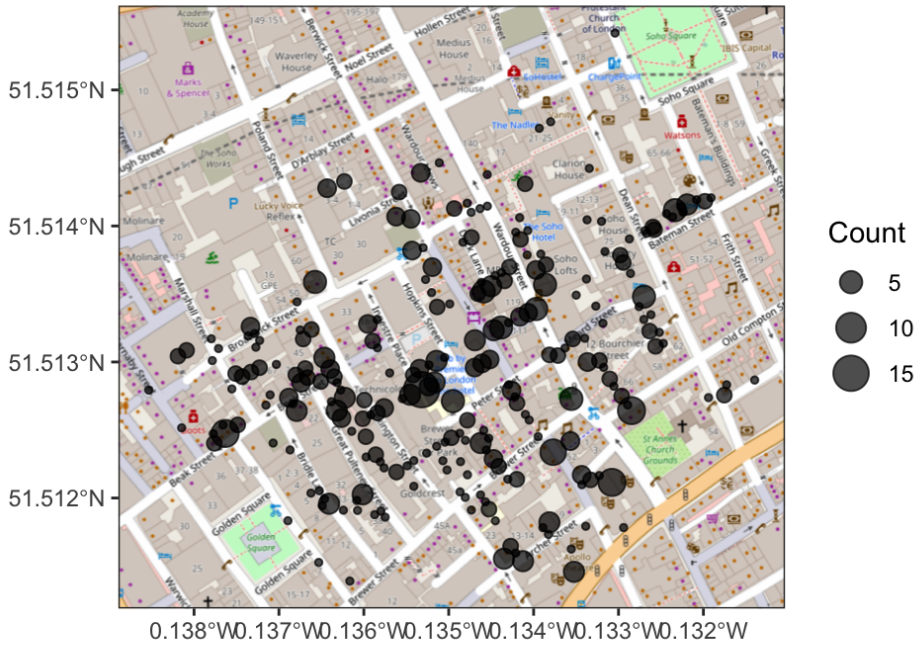
For all our previous maps, we only have partial context for what we are seeing. For a more integrative view, we may want to overlay our map over *Google Maps-like* physical features.

For example, for our London cholera outbreak Dot map, we want overlay it on the London street map–and this is exactly what `{ggspatial}` lets us do.

The `annotation_map_tile()` function adds a layer of **map tiles** pulled from Open Street Map. We can control the `zoom` level, as well as the `type`. Here, we map the number of deaths at each location to the size of the dot.

```
p <-
  ggplot(mdsr::CholeraDeaths) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(mapping = aes(size = Count), alpha = 0.7)
```

```
print(p)
```



RSTUDIO CLOUD

Add a Basemap to a Dot map using the `africapitals` object and the `annotation_map_tile()` function.

WATCH OUT

If you are publishing a map using these tiles, make sure to use the proper attribution (e.g., "Copyright OpenStreetMap contributors" when using an OpenStreetMap-based tile set)

19.7 Wrap up

In this lesson, we learned about *Thematic maps*, how to create them using *{ggplot2}* and the `geom_sf()` function, and which type of *Spatial data* they visualize.

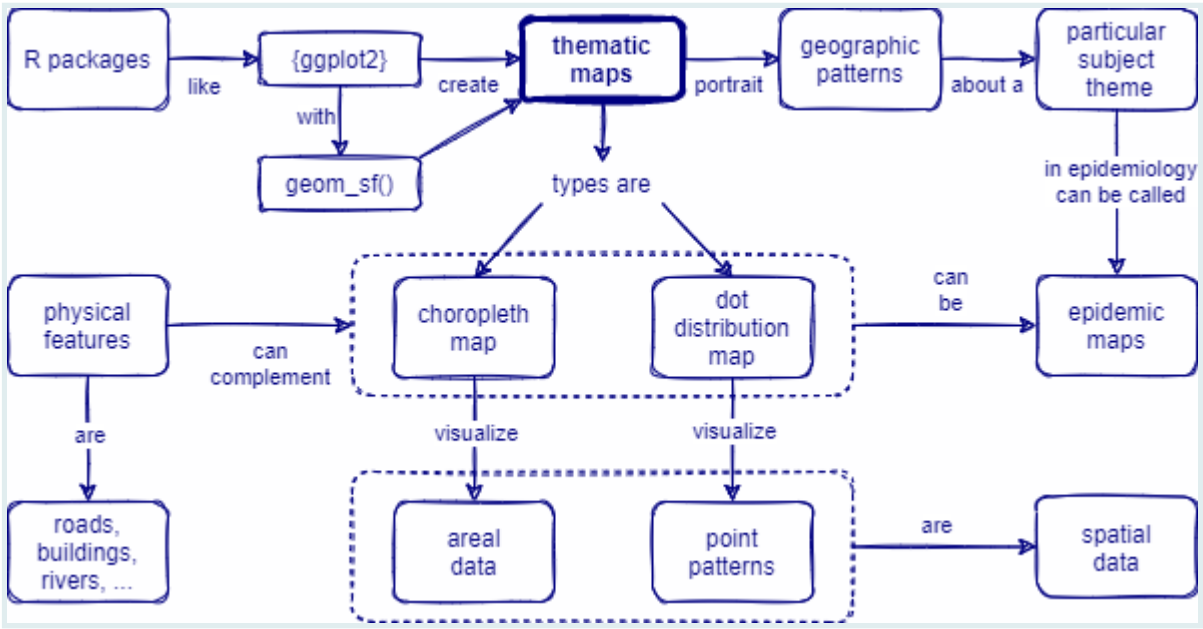


Figure 4. Concept map #1.

But, How can we create more *Thematic maps* from **external** Spatial data? In the next lesson, we are going to learn how to **read external Spatial data** from *online* repositories and *local* files!

Contributors

The following team members contributed to this lesson:

ANDREE VALLE CAMPOS
R Developer and Instructor, The Graph Network
Motivated by reproducible science and education.

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- *Batra, Neale, et al. (2021). The Epidemiologist R Handbook. Chapter 28: GIS Basics*. (2021). Retrieved 01 April 2022, from https://epirhandbook.com/en/gis-basics.html
- *Lovelace, R., Nowosad, J., & Muenchow, J. Geocomputation with R. Chapter 2: Geographic data in R*. (2019). Retrieved 01 April 2022, from https://geocompr.robinlovelace.net/spatial-class.html
- *Moraga, Paula. Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny. Chapter 2: Spatial data and R packages for mapping*. (2019). Retrieved 01 April 2022, from https://www.paulamoraga.com/book-geospatial/sec-spatialdataandCRS.html
- *Baumer, Benjamin S., Kaplan, Daniel T., and Horton, Nicholas J. Modern Data Science with R. Chapter 17: Working with geospatial data*. (2021). Retrieved 05 June 2022, from https://mdsr-book.github.io/mdsr2e/ch-spatial.html

20

Geospatial analysis: Read external data

20.1 Learning objectives

1. Read Spatial data from **online databases** using `{rgeoboundaries}` and `{malariaAtlas}`.

2. Convert foreign objects to **`sf` class** using the `st_as_sf()` function from the `{sf}` package.

3. Relate each Thematic map with a **Vector data** type.

20.2 Prerequisites

This lesson requires the following packages:

```
if(!require('pacman')) install.packages('pacman')

pacman::p_load(malariaAtlas,
               ggplot2,
               cholera,
               here,
               sf)

pacman::p_load_gh("afrimapr/afrilearndata",
                  "wmgeolab/rgeoboundaries")
```

20.3 Introduction

In the previous lesson we learned how to create *Thematic maps* using `{ggplot2}` and the `geom_sf()` function.

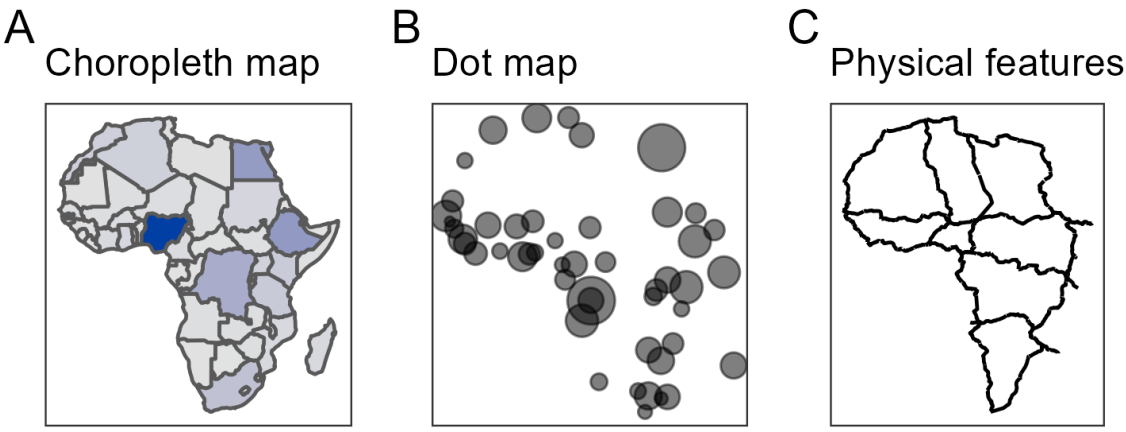


Figure 1. (A) Estimated population of each country in the African continent.; (B) Number of inhabitants in each capital city of the African continent. (C) African trans-continental highway network lines.

For all of our previous examples we used *Spatial data* recorded inside the `{afrilearndata}` package, like `africapitals`, `africountries`, and `afrihighway`. But, How can we create more *Thematic maps* from **external** Spatial data?

In this lesson we are going to learn how to **read external Spatial data** from *online* repositories.

20.4 From Online databases

Here we are going to explore two `{packages}` that **connect** our environment to web repositories and facilitate the downloading of external data. This procedure *do not need* URL paths, only specific *arguments* inside a function.

20.4.1 Boundaries of countries with `rgeoboundaries`

The `rgeoboundaries` package is a client for the geoBoundaries API, providing country political administrative boundaries.

A single country

To download boundaries of countries we use the `geoboundaries()` function of `{rgeoboundaries}`. For example, we can download the administrative boundary of Zimbabwe and assign it to a variable called `zimbabwe_boundary` as follows.

```
zimbabwe_boundary <- geoboundaries(country = "Zimbabwe")
```

- The `zimbabwe_boundary` is a `"sf"` class object.
- `{ggplot2}` allows us to easily visualise **simple feature** objects using the `geom_sf()` function.
- It can be used to plot the administrative boundary of Zimbabwe as follows:

```
ggplot(data = zimbabwe_boundary) +
  geom_sf()
```

Download the boundaries of `Sierra Leone`.

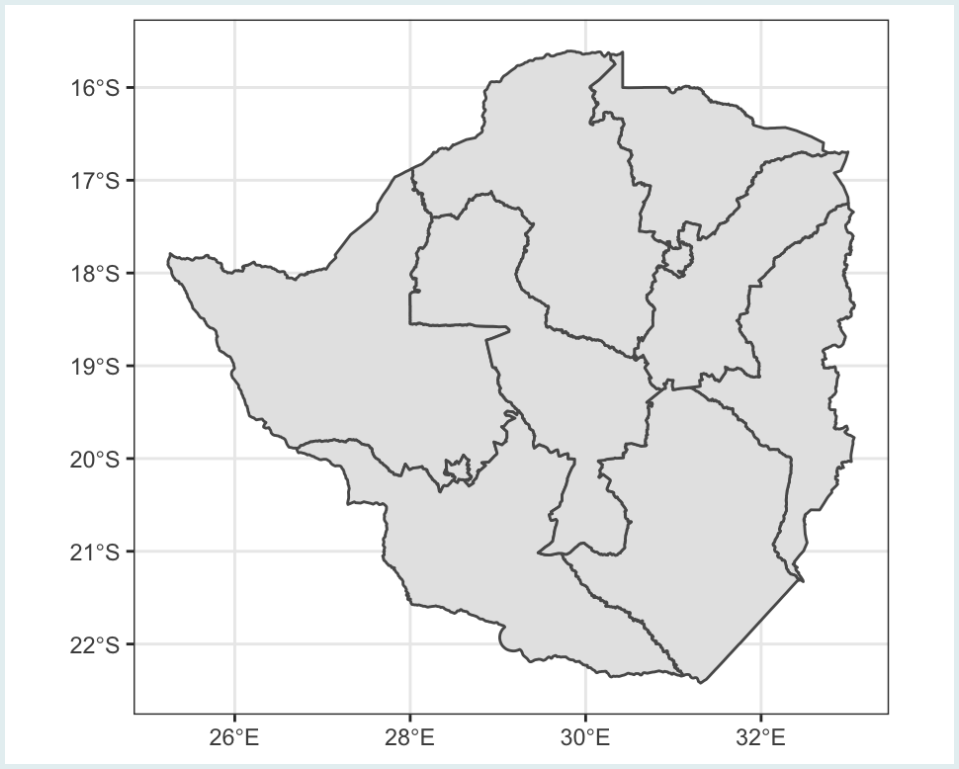
Different levels of administrative boundaries

If available, lower levels of administrative boundaries in countries can be downloaded too. We just have to pass the administrative level as an argument in the `geoboundaries()` function.

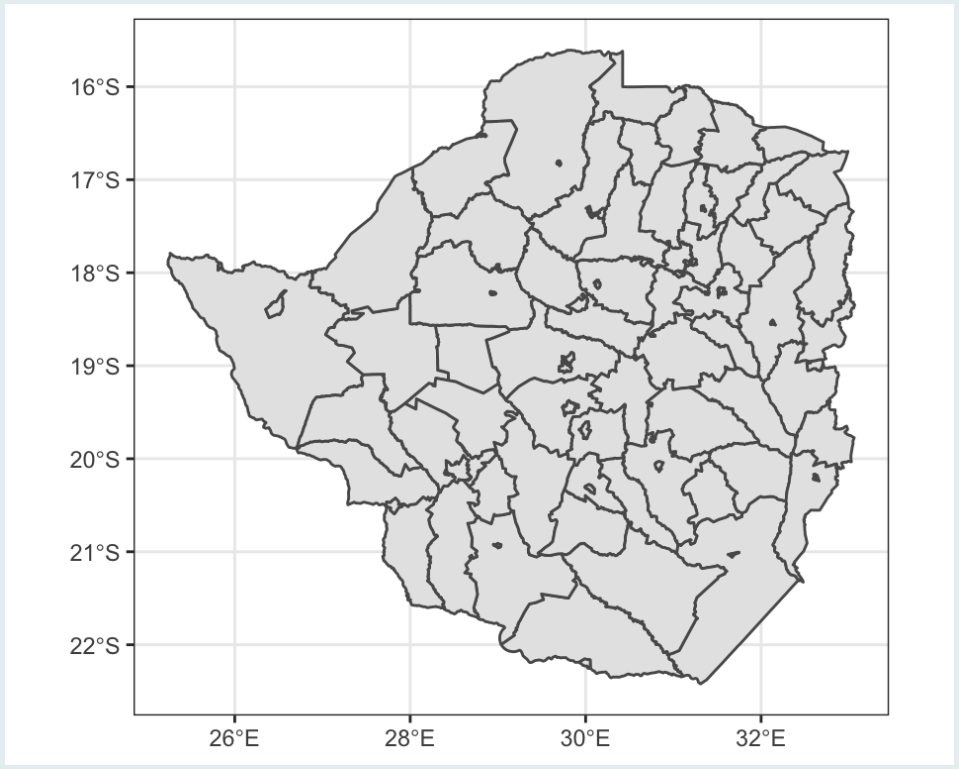
Administrative **level 1** (`"adm1"`) is the highest level, while administrative **level 5** (`"adm5"`) is the lowest. This means the country will be further sub-divided into administrative divisions as the Administrative level progresses from 1 to 5.

See how the first and second administrative level boundaries of Zimbabwe are downloaded below.

```
# downloading administrative level 1 boundaries
zimbabwe_boundaries_adm1 <- geoboundaries(country = "Zimbabwe",
                                          adm_lvl = "adm1")

ggplot(data = zimbabwe_boundaries_adm1) +
  geom_sf()
```




```
# downloading administrative level 2 boundaries
zimbabwe_boundaries_adm2 <- geoboundaries(country = "Zimbabwe",
                                          adm_lvl = "adm2")

ggplot(data = zimbabwe_boundaries_adm2) +
  geom_sf()
```



Download the `third` administrative level boundaries of `Sierra Leone`.

We can also download the boundaries of **multiple countries** together by including the names of countries as a `vector` class object like: `c("country_01","country_02")`.

See how the `second` administrative level boundaries of adjacent countries like `Zimbabwe` and `Mozambique` are downloaded and plotted below.

```
zimbabwe_mozambique_adm2 <-
  geoboundaries(country = c("Zimbabwe", "Mozambique"),
                                    adm_lvl = "adm2")
```

```
ggplot(data = zimbabwe_mozambique_adm2) +
  geom_sf()
```

20.4.2 Disease information with `malariaAtlas`

The `malariaAtlas` package download, visualize and manipulate global malaria data hosted by the Malaria Atlas Project.

The `malariaAtlas` package enables users to download data like:

- parasite rate (PR) survey data (*Plasmodium falciparum* and *Plasmodium vivax*)
- mosquito occurrence data

Parasite Rate surveys

The `getPR()` function downloads all the publicly available PR points for a country (or countries) and returns it as a dataframe. The `species` argument is a string specifying the Plasmodium species and can be `Pf` (*Plasmodium falciparum*), `Pv` (*Plasmodium vivax*) or `BOTH`.

```
zimbabwe_malaria_pr <- getPR(country = "Zimbabwe", species = "BOTH")
```

The output is a `data.frame` object that contains `longitude` and `latitude` as variables:

```
class(zimbabwe_malaria_pr)
```

```
## [1] "pr.points"  "data.frame"
```

```
zimbabwe_malaria_pr %>%
  as_tibble()
```

```
## # A tibble: 567 × 28
##    dhs_id site_id site_name    latitude longitude rural_urban country
country_id continent_id month_start year_start month_end year_end lower_age
##    <lgl>    <int> <chr>           <dbl>     <dbl> <chr>       <chr>
<chr>      <chr>               <int>      <int>     <int>    <int>     <int>
##  1 NA       16142 Chilonga        -18.4      27.2 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  2 NA       17018 Ume - Kahobo    -17.8      28.8 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  3 NA       18443 Bandanyenje     -18.9      32.0 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  4 NA          68 Piringani       -17.0      30.1 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  5 NA       19390 Zidulini        -18.7      28.7 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  6 NA       17564 Lutumba         -22.1      30.2 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  7 NA       14287 Seula School    -21.6      28.4 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  8 NA        2943 Ndlovu          -18.9      27.8 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
##  9 NA       11029 Bambaninga      -20.4      31.8 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
## 10 NA       20058 Pumula Miss…    -19.6      27.1 UNKNOWN     Zimbab… ZWE
Africa                 3       1991         4      1991         4
## # … with 557 more rows, and 14 more variables: upper_age <int>, examined
<int>, positive <int>, pr <dbl>, species <chr>, method <chr>,
## #   rdt_type <chr>, pcr_type <lgl>, malaria_metrics_available <chr>,
location_available <chr>, permissions_info <chr>, citation1 <chr>,
## #   citation2 <chr>, citation3 <chr>
```

PRACTICE

(in RMD)

Download the publicly available Parasite Ratio (PR) points for *Plasmodium falciparum* in Sierra Leone.

`autoplot()` can be used to quickly and easily visualise the downloaded PR survey points.

```
autoplot(zimbabwe_malaria_pr)
```

However, if you would like to make a **custom plot** for your particular research needs, with complete control of the *output aesthetics*, you would probably prefer `{ggplot2}` for the work!

Convert a foreign object to `sf`

To convert a `data.frame` to an `sf` object, we can use the `st_as_sf()` from the `{sf}` package.

In case of **Point** patterns data (like for a **Dot map**):

- the `coords` argument need to specify the names of the variables holding the **Coordinates**, and
- the `crs` argument specify the **Coordinate Reference System (CRS)** to be assigned (here WGS84, which is the CRS code #`4326`).

```
zimbabwe_malaria_pr_sf <-
  # data frame
  zimbabwe_malaria_pr %>%
  # convert to sf
  sf::st_as_sf(coords = c("longitude","latitude"),
               crs = 4326)
```

Inside the **`coords` argument**,you need to pay attention to two things:

- You need to use `"quotation_marks"` for each column name,
- First write the `longitude`, and then the `latitude`.

Does this CRS business sound like gibberish to you? DON'T PANIC ! We are going to cover this SOON in the coming lessons. We will see what "WGS84" stands for and how to choose the right CRS for our plots. For now, trust me on this.

Mission accomplished! This is the output of the convertion:

```
zimbabwe_malaria_pr_sf
```

```
## Simple feature collection with 567 features and 26 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 25.8063 ymin: -22.2333 xmax: 32.9708 ymax: -16.0662
## Geodetic CRS:  WGS 84
## First 10 features:
##    dhs_id site_id      site_name rural_urban  country country_id
continent_id month_start year_start month_end year_end lower_age upper_age
## 9      NA   16142       Chilonga     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 11     NA   17018   Ume - Kahobo     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 17     NA   18443    Bandanyenje     UNKNOWN Zimbabwe        ZWE
```

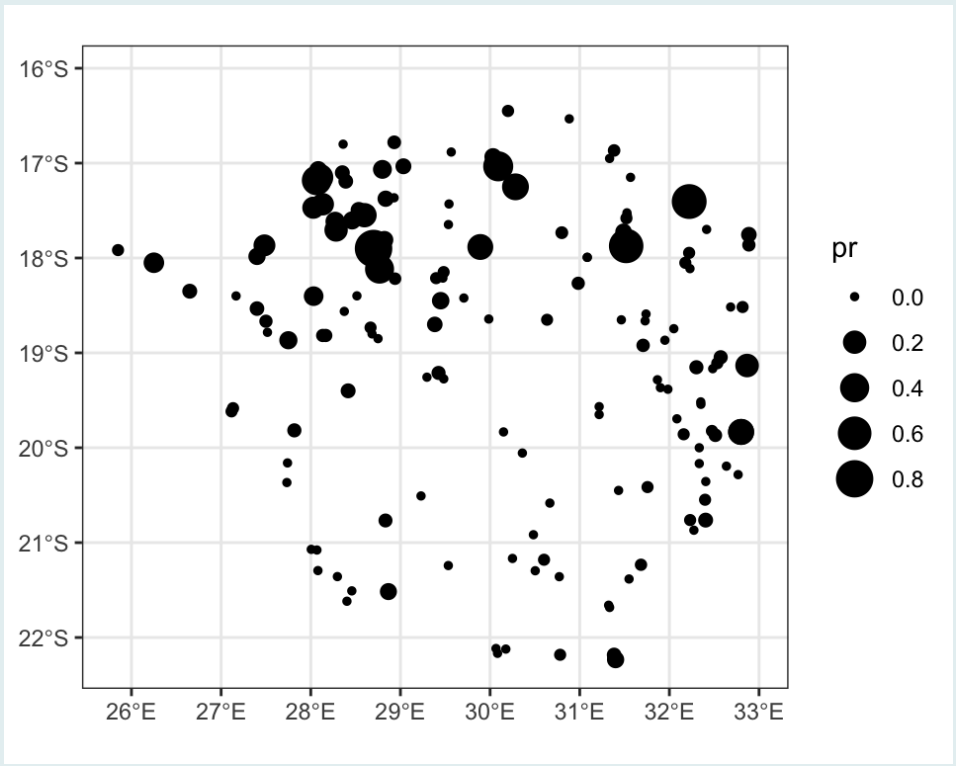
```
Africa           3       1991         4     1991         4        16
## 19     NA      68      Piringani     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 21     NA   19390       Zidulini     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 24     NA   17564        Lutumba     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 26     NA   14287   Seula School     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 30     NA    2943         Ndlovu     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 34     NA   11029     Bambaninga     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
## 38     NA   20058 Pumula Mission     UNKNOWN Zimbabwe        ZWE
Africa           3       1991         4     1991         4        16
##    examined positive     pr       species     method rdt_type pcr_type
malaria_metrics_available location_available permissions_info
## 9       107        0 0.0000 P. falciparum Microscopy                NA
true               true
## 11       40        3 0.0750 P. falciparum Microscopy                NA
true               true
## 17      125        0 0.0000 P. falciparum Microscopy                NA
true               true
## 19      103        0 0.0000 P. falciparum Microscopy                NA
true               true
## 21      125        1 0.0080 P. falciparum Microscopy                NA
true               true
## 24      118        0 0.0000 P. falciparum Microscopy                NA
true               true
## 26      118        0 0.0000 P. falciparum Microscopy                NA
true               true
## 30      125       10 0.0800 P. falciparum Microscopy                NA
true               true
## 34      125        1 0.0080 P. falciparum Microscopy                NA
true               true
## 38      129        1 0.0078 P. falciparum Microscopy                NA
true               true
##
citation1
## 9  Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 11 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 17 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 19 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 21 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 24 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
```

```
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 26 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 30 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 34 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
## 38 Freeman, T.W. and Chandiwana, S.K. (1996).  <b>A retrospective analysis
of data from the 1991 national malaria prevalence survey in Zimbabwe.</b>
<i>Malaria and Infectious Diseases in Africa</i>, <b>4</b>():19-24
##                                                 citation2
## 9   (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 11  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 17  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 19  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 21  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 24  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 26  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 30  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 34  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
## 38  (2010). <b>http://mara-database.org/mara/.</b> . MARA/ARMA.
##
citation3
## 9  Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 11 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 17 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 19 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 21 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 24 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 26 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 30 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 34 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
## 38 Le Sueur, D., Binka, F., Lengeler, C., De Savigny, D., Snow, B.,
Teuscher, T. and Toure, Y. (1997).  <b>An atlas of malaria in Africa.</b>
<i>Africa health</i>, <b>19</b>(2):23-4
```

```
##                     geometry
## 9     POINT (27.1667 -18.4)
## 11 POINT (28.8209 -17.8091)
## 17   POINT (31.95 -18.8667)
## 19 POINT (30.1333 -16.9667)
## 21 POINT (28.6667 -18.7333)
## 24  POINT (30.1757 -22.122)
## 26 POINT (28.4033 -21.6176)
## 30   POINT (27.75 -18.8667)
## 34 POINT (31.7562 -20.4146)
## 38   POINT (27.116 -19.617)
```

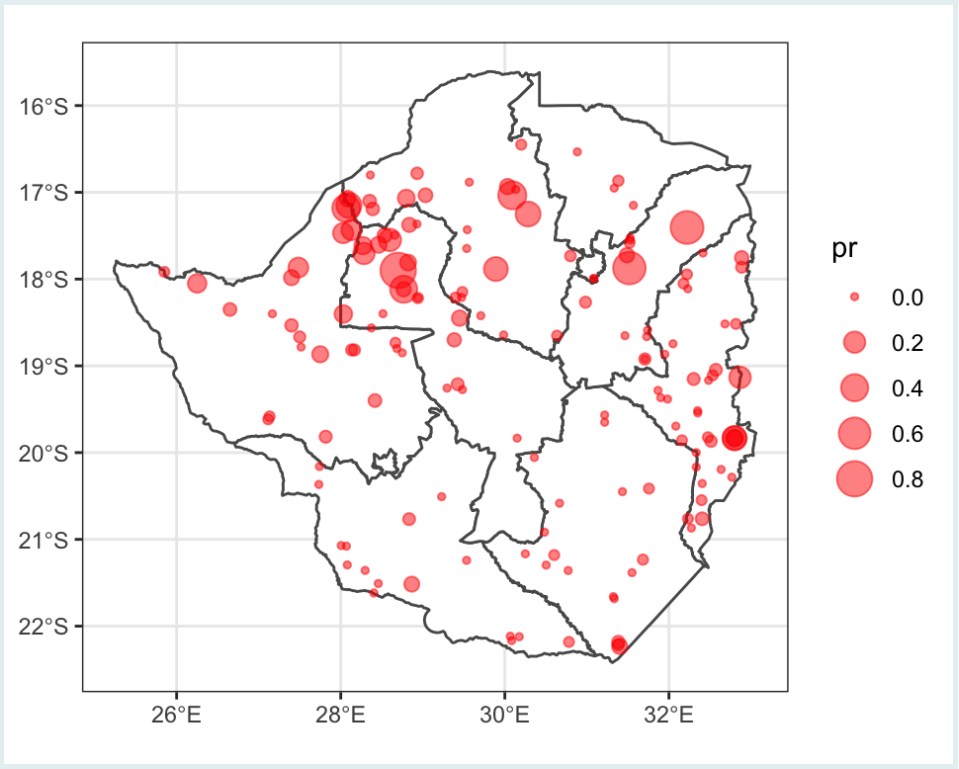
Now we have converted our data into an `sf` object and can directly plot this object with `{ggplot2}` code:

```
ggplot(data = zimbabwe_malaria_pr_sf) +
  geom_sf(aes(size = pr))
```



Or add one previous layer with the administrative boundaries of Zimbabwe:

```
ggplot() +
  # boundaries data
  geom_sf(data = zimbabwe_boundaries_adm1,
          fill = "white") +
  # disease data
  geom_sf(data = zimbabwe_malaria_pr_sf,
          aes(size = pr),
          color = "red",
          alpha=0.5)
```



`sf::st_as_sf()` does not allow missing values in coordinates!

So you can use `{dplyr}` verbs like `filter()` to drop problematic rows from variable `x` using this notation:`filter(!is.na(x))`.

With the publicly available Parasite Ratio (PR) points of *Plasmodium falciparum* in Sierra Leone stored in `q3`:

Convert its `data.frame` output to a `sf` object using the code `4326` to set a WGS84 CRS. (More on all these codes coming soon, don't be intimidated !)

Vector data

Vector Data is the most common format of *Spatial data* used in GIS. *Vector data* provides sufficient information about where different features should be, in relation to one another.

- When we collect *Point* pattern data, used to create *Dot maps*, we need to use a **Vector data** type called **Point** geometry.
- When we collect *Areal* data, used to create *Choropleth maps*, we need to use a **Vector data** type called **Polygon** geometry.

Above is an example of converting foreign **Point** data to `sf` class. Now let's see briefly see how to convert foreign **Areal** data.

Here is an example, using an alternative way to download administrative boundaries using `raster::getData()``:

1. We get *level 2* administrative boundaries data from the GADM online repository.
2. Countries are specified by their 3 letter ISO codes.

```
pacman::p_load(raster)
sle_polygon <- getData(name = 'GADM', country = 'SLE', level = 2)
sle_polygon
```

```
## class       : SpatialPolygonsDataFrame
## features    : 14
## extent      : -13.30351, -10.26575, 6.917616, 10.00043  (xmin, xmax, ymin,
ymax)
## crs         : +proj=longlat +datum=WGS84 +no_defs
## variables   : 13
## names       : GID_0,       NAME_0,   GID_1,  NAME_1, NL_NAME_1,     GID_2,
NAME_2, VARNAME_2, NL_NAME_2,   TYPE_2, ENGTYPE_2, CC_2,   HASC_2
## min values  :   SLE, Sierra Leone, SLE.1_1, Eastern,         NA, SLE.1.1_1,
Bo,  Freetown,        NA,     Area,      Area,   NA, SL.EA.KE
## max values  :   SLE, Sierra Leone, SLE.4_1, Western,         NA, SLE.4.2_1,
Western Urban,  Freetown,        NA, District,  District,   NA, SL.WE.WU
```

3. Countries are *areas* (i.e. Spatial **Polygons**) hence we need to convert these area coordinates to an `sf` object.

```
sle_polygon %>% class()
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

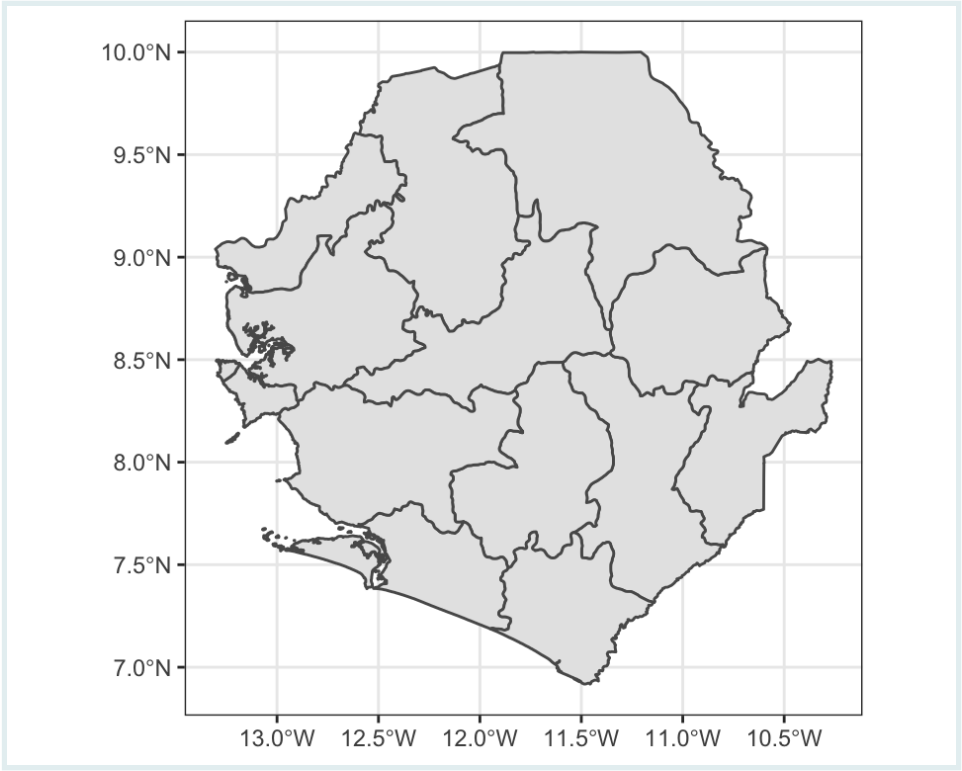
4. Let's see how this translated into code (reminder: you do not need to use any argument inside `st_as_sf()`):

```
sle_polygon %>% sf::st_as_sf() %>% class()
```

```
## [1] "sf"          "data.frame"
```

5. With this `sf` object we can easily make plots with {ggplot2}

```
sle_polygon %>%
  sf::st_as_sf() %>%
  ggplot() +
  geom_sf()
```



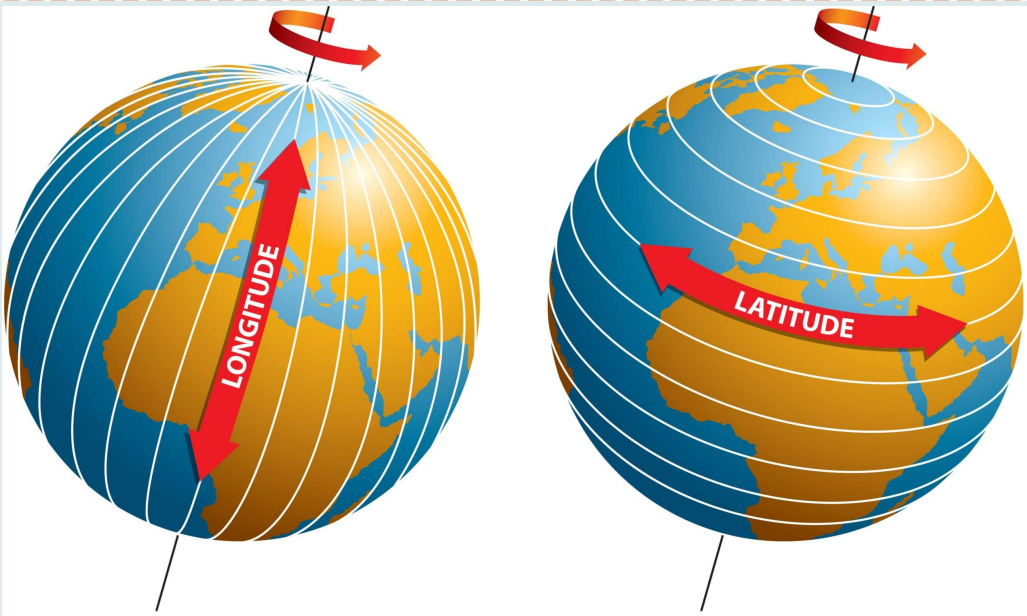
WATCH OUT

Do not get confused by the `vector` class object, which is an R class just like `data.frame` and `matrix`.

VOCAB

Vector data require a **Coordinate Reference System (CRS)** to relate the spatial elements of the data with the *surface of Earth*.

Degrees of **longitude/latitude** are the most common.



For now, it is sufficient to know that coordinate systems are a *key component* of geographic objects. We will cover them in detail later !

20.5 Wrap up

In this lesson, we have learned how to **read** Vector data like *Polygon geometries* from *Online databases* from the `{rgeoboundaries}` package and `GADM` repository, how to **convert** foreign *Point geometries* to `sf` from the `{malariaAtlas}` package, and why they need a *Coordinate Reference Systems (CRS)*.

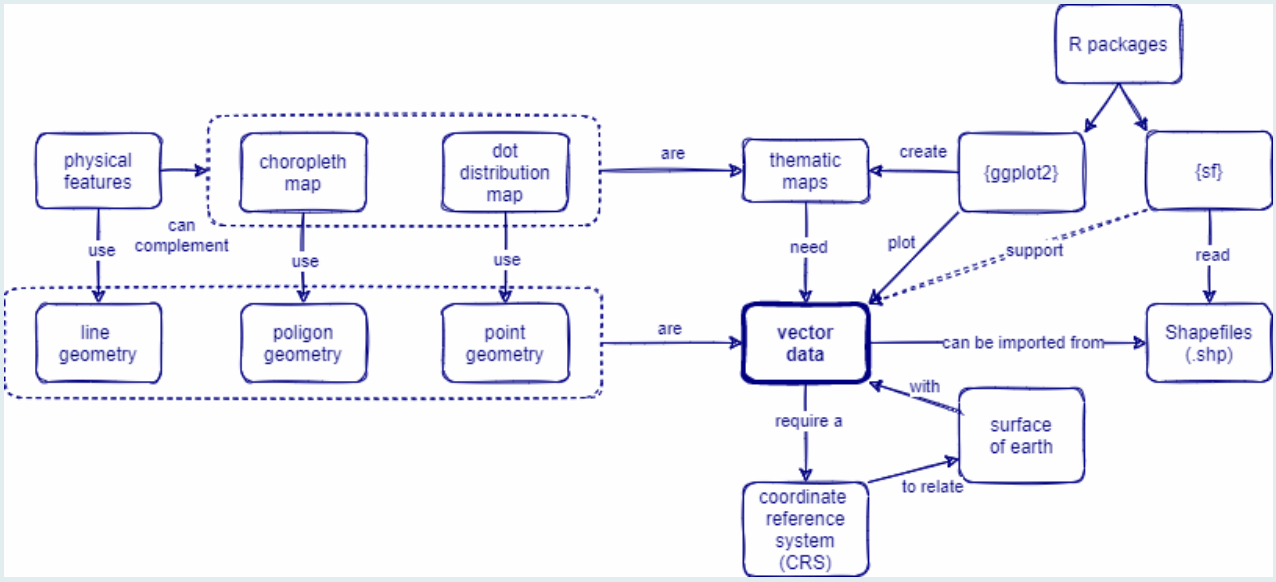


Figure 5. Concept map

In the next lesson we are going dive into **Vector data** components and how to read it from **local files**!

Contributors

The following team members contributed to this lesson:

ANDREE VALLE CAMPOS

R Developer and Instructor, The Graph Network
Motivated by reproducible science and education.

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- *Seimon, Dilinie. Administrative Boundaries.* (2021). Retrieved 15 April 2022, from https://rspatialdata.github.io/admin_boundaries.html

- *Varsha Ujjinni Vijay Kumar. Malaria.* (2021). Retrieved 15 April 2022, from https://rspatialdata.github.io/malaria.html
- *Batra, Neale, et al. The Epidemiologist R Handbook. Chapter 28: GIS Basics*. (2021). Retrieved 01 April 2022, from https://epirhandbook.com/en/gis-basics.html
- *Lovelace, R., Nowosad, J., & Muenchow, J. Geocomputation with R. Chapter 2: Geographic data in R*. (2019). Retrieved 01 April 2022, from https://geocompr.robinlovelace.net/spatial-class.html
- *Moraga, Paula. Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny. Chapter 2: Spatial data and R packages for mapping*. (2019). Retrieved 01 April 2022, from https://www.paulamoraga.com/book-geospatial/sec-spatialdataandCRS.html

 # Geospatial analysis: Read local data

```
title: ' ' output: # word_document: # toc: true html_document: number_sections: true toc: true css: !expr here::here("global/style/style.css") highlight: kate pandoc_args: -shift-heading-level-by=-1 editor_options: markdown: wrap: 100 canonical: true chunk_output_type: inline
```

21

Geospatial analysis: coordinate reference systems

21.1 Learning objectives

1. Zoom in `ggplot` maps with Coordinate Reference Systems (**CRS**) using the `coord_sf()` function.
2. Change the **CRS projection** of a `ggplot` map using the `crs` argument of `coord_sf()`.
3. Configure data with **UTM coordinate system** using the `st_as_sf()` function.
4. Change the **CRS projection** of a `sf` object using the `st_transform()` function.

21.2 Prerequisites

This lesson requires the following packages:

```
if(!require('pacman')) install.packages('pacman')

pacman::p_load(malariaAtlas,
               colorspace,
               ggplot2,
               cholera,
               spData,
               here,
               rio,
               sf)

pacman::p_load_gh("afrimapr/afrilearndata",
                  "wmgeolab/rgeoboundaries")
```

21.3 Introduction

From the second lesson, we learned that **Vector data** require a **Coordinate Reference System (CRS)** to relate the spatial elements of the data with the *surface of Earth*. For that reason, Coordinate systems are a *key component* of geographic objects.

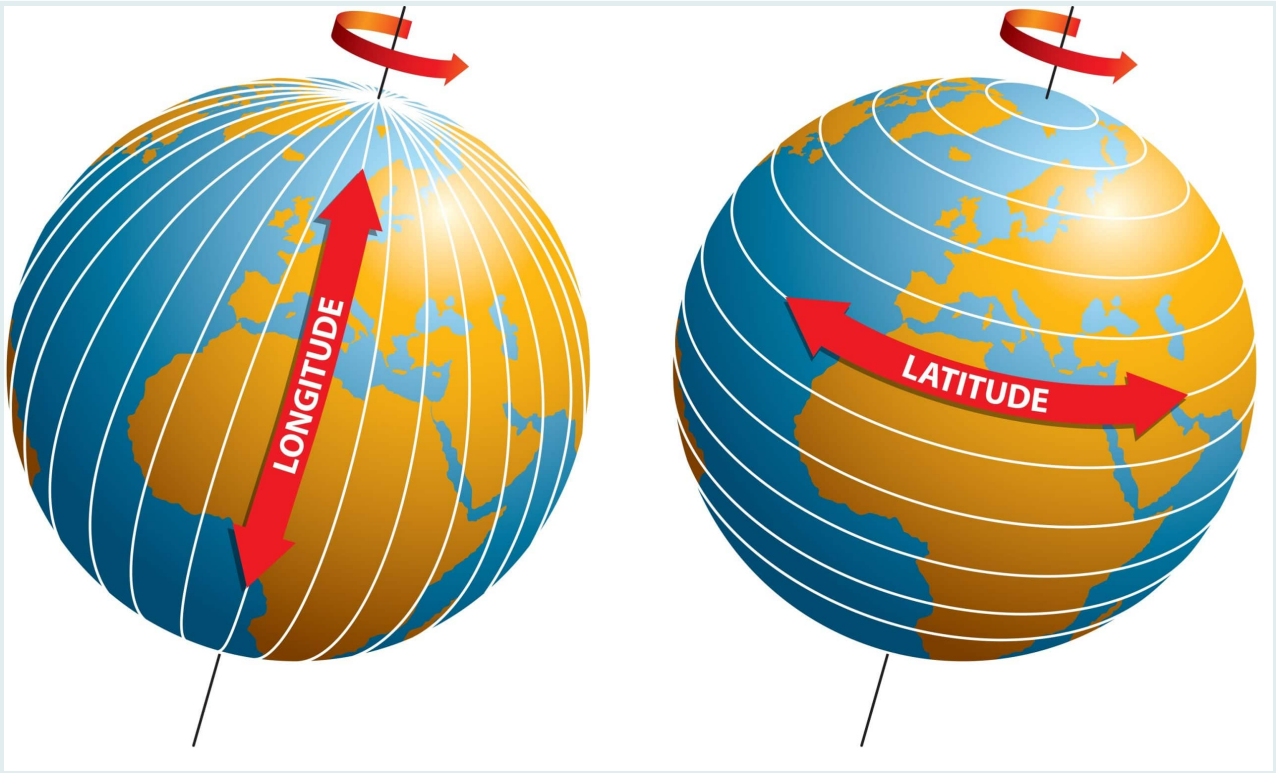


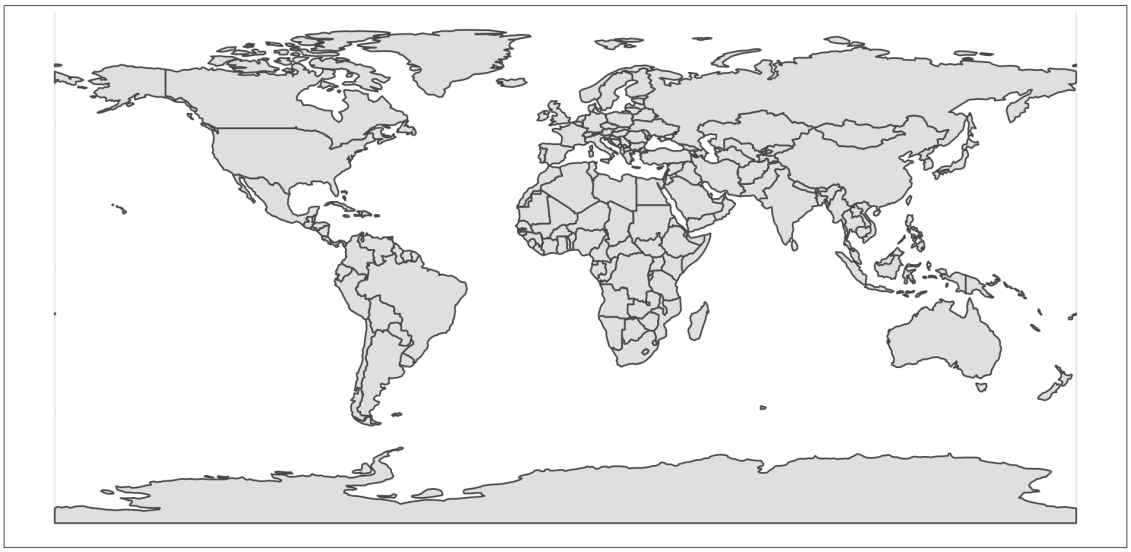
Figure 1. Direction of longitude and latitude.

In this lesson we are going to learn how to manage the CRS of maps by **zooming in** to an area of interest, **set them up** to external data with coordinates different to longitude and latitude (called UTM), and **transform** between different coordinate systems of CRS's.

21.4 Data and basic plot

First, let us start with creating a base map of the `world` from the `spData` package using `ggplot2`:

```
ggplot(data = world) +
    geom_sf()
```



21.5 Manage Coordinate systems with `coord_sf()`

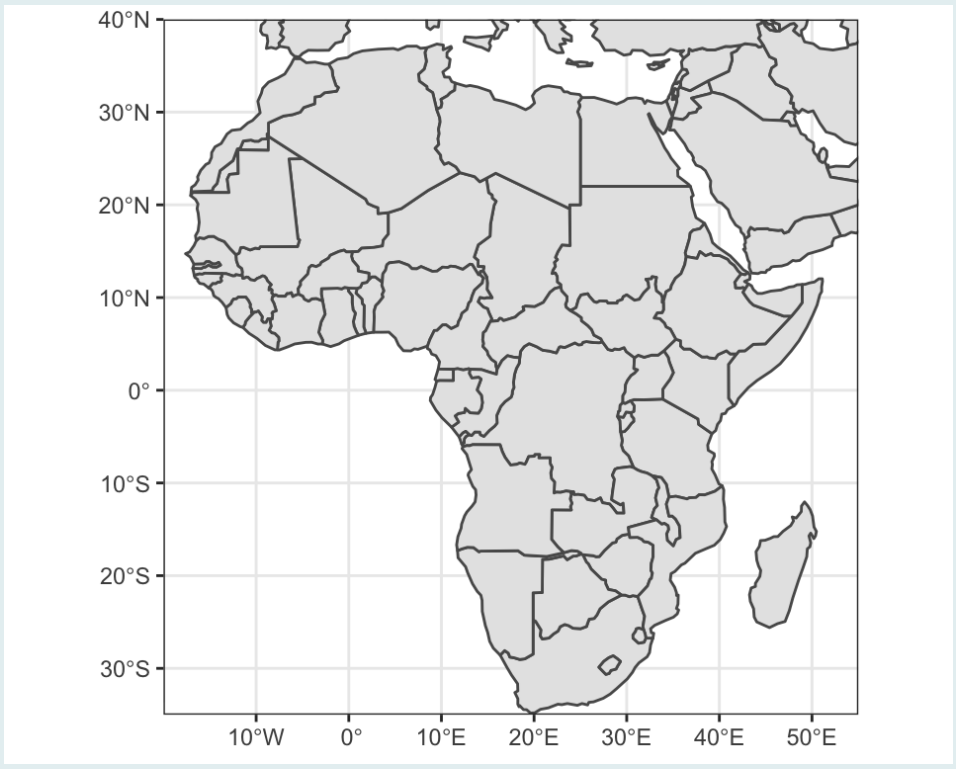
The function `coord_sf()` from the `{ggplot2}` package allows to deal with the **coordinate system**, which includes both the *extent* and *projection* of a map.

21.5.1 "Zoom in" on maps

The **extent** of the map can also be set in `coord_sf()`, in practice allowing to "zoom" in the area of interest, provided by limits on the x-axis (`xlim`), and on the y-axis (`ylim`).

Here, we zoom in the `world` map to the African continent, which is in an area delimited in **longitude** between 20°W and 55°E, and in **latitude** between 35°S and 40°N. To exactly match the limits provided, use `expand = FALSE`.

```
ggplot(data = world) +
    geom_sf() +
    coord_sf(xlim = c(-20, 55), ylim = c(-35, 40), expand = FALSE)
```



Check which + and – **signs** are related with the **cardinal direction**:

- In longitude: West is –, East is +,
- In latitude: South is –, North is +.

Zoom in the `africountries` map to Sierra Leona, which is in an area delimited in longitude between 14°W and 10°W, and in latitude between 6°N and 10°N.

21.5.2 Change the Projection of a map

The `world` object have a **CRS projection** called *WGS84* (detailed in the **fifth line** of the **header**)

```
world
```

```
## Geometry set for 177 features
## Geometry type: MULTIPOLYGON
## Dimension:     XY
```

```
## Bounding box:  xmin: -180 ymin: -89.9 xmax: 180 ymax: 83.64513
## Geodetic CRS:  WGS 84
## First 5 geometries:
```

Which corresponds to the **EPSG code 4326**:

```
st_crs(world)$input
```

```
## [1] "EPSG:4326"
```

KEY POINT

Projection refers to the mathematical equation that was used to *project* the truly *round* earth onto a *flat* surface.

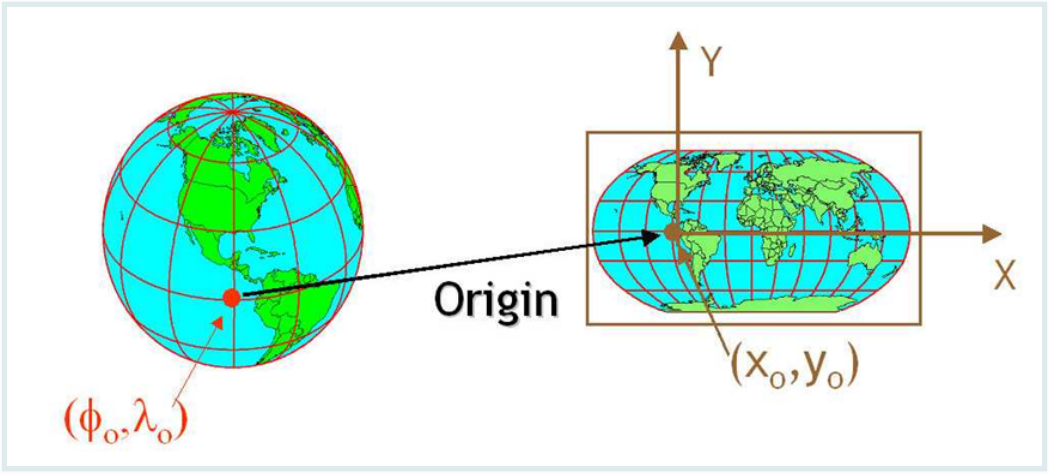


Figure 2. Projection of a CRS.

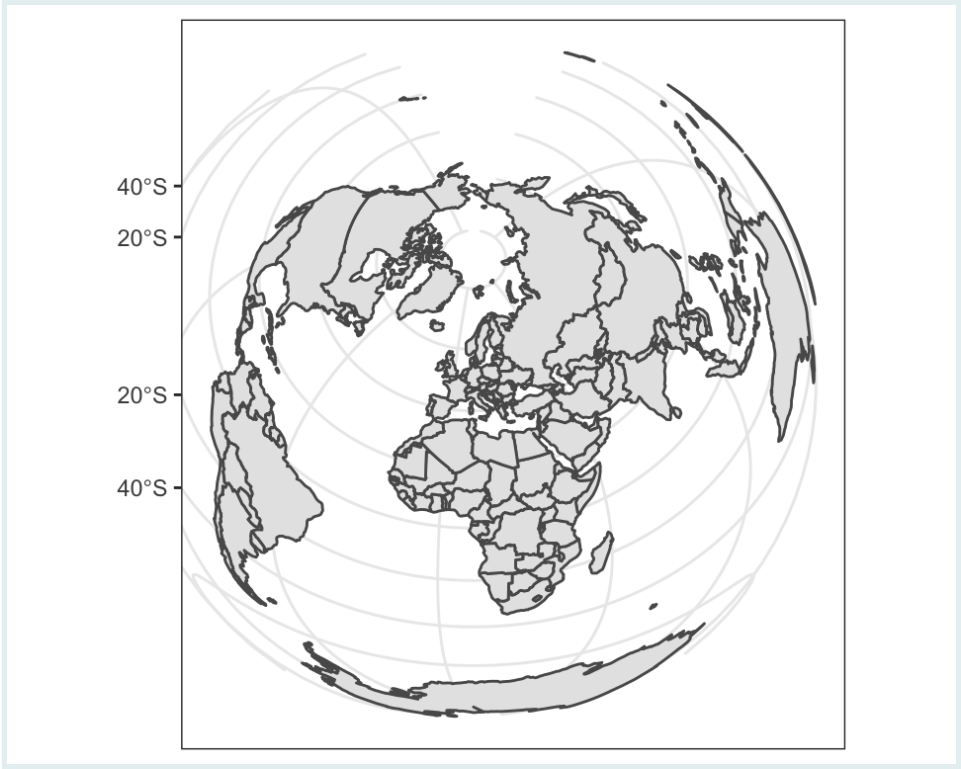
VOCAB

- **EPSG** refers to the *European Petroleum Survey Group (EPSG)*.
- *EPSG* is a Spatial Reference System Identifier (SRID) with arbitrary **codes** available for concrete **CRS projections**.
- One of these projections is **WGS84**, which refers to *World Geodetic System 1984*.

Using the `crs` argument of the `coord_sf()` function, it is possible to **override** this setting, and project on the fly to **any projection**.

For example, we can change the current *WGS84* projection to the *ETRS89 Lambert Azimuthal Equal-Area projection* (alias LAEA), which is **EPSG code 3035**:

```
ggplot(data = world) +
    geom_sf() +
    coord_sf(crs = 3035)
```



However, this CRS projection is useful for European countries.

PRACTICE (in RMD)

Get into the linked webpage to find the *EPSG code* that its required:

Change the CRS projection of the `ggplot` map with the `world` object to the Pseudo-Mercator coordinate system.

Choosing a Projection / CRS

REMINDER

Which projection should I use?

To decide if a projection is right for your data, answer these questions:

- What is the area of minimal distortion?

REMINDER

- What aspect of the data does it preserve?

PRO TIP

Take the time to identify a **projection** that is suited for your project. You don't have to stick to the ones that are popular.

Online tools like Projection Wizard can also help you **discover** projections that might be a **better fit for your data**.

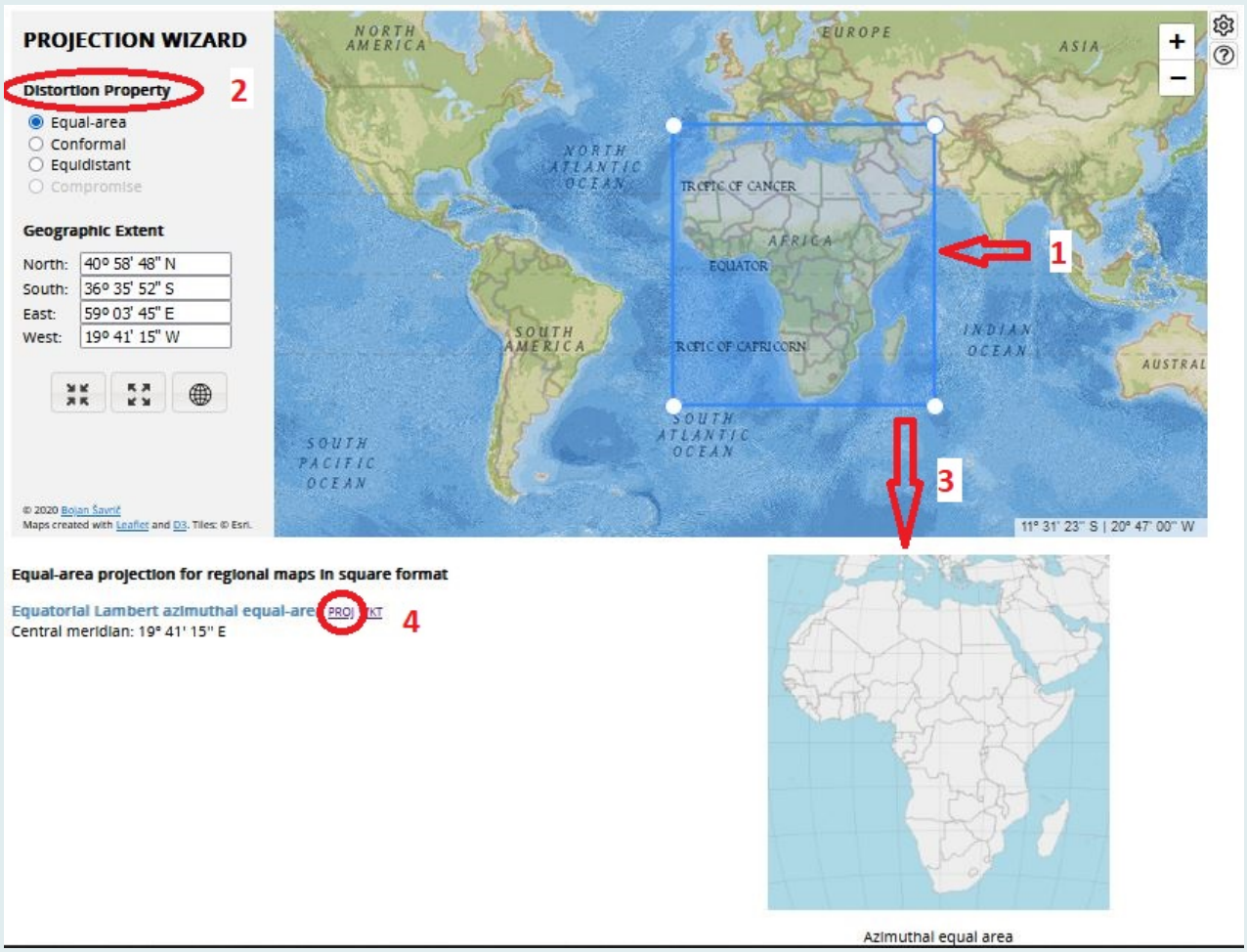


Figure 3. Steps to find a custom projection with Project Wizard.

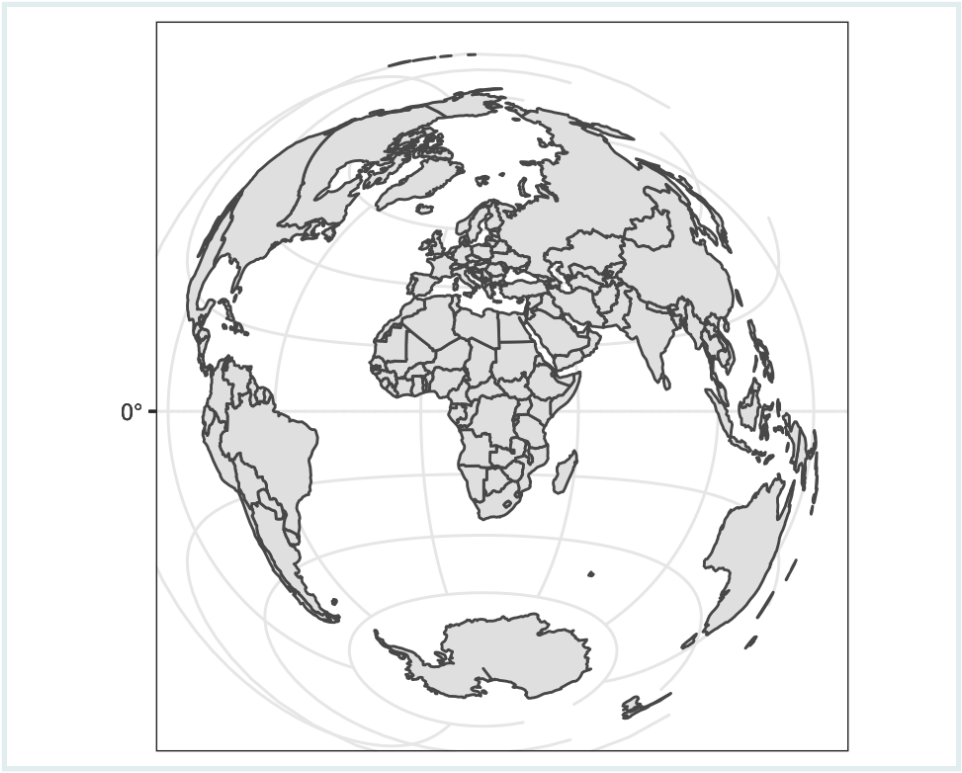
For instance, to **find** an appropriate projection for the African continent, you can:

1. Define your *area* of interest,
2. Select a *distortion* property,
3. Confirm the map outcome that *fits* your needs,

4. Copy the text inside the **PROJ** option.

Then, paste that valid **PROJ string** to the `crs` argument:

```
ggplot(data = world) +
    geom_sf() +
    coord_sf(crs = "+proj=laea +lon_0=19.6875 +lat_0=0 +datum=WGS84 +units=m
        +no_defs")
```



PROJ is an open-source library for storing, representing and transforming CRS information.

Get into the linked webpage to find the *PROJ string* that its required:

Change the CRS projection of the `ggplot` map with the `world` object to the Aitoff coordinate system.

CRS components

A **CRS** has a few key components:

Coordinate System - There are many many different coordinate systems, so make sure you know which system your coordinates are from. (e.g. longitude/latitude, which is the most common);

- *Units* - Know what the units are for your coordinate system (e.g. decimal degrees, meters);
- *Datum* - A particular modeled version of the Earth. These have been revised over the years, so ensure that your map layers are using the same datum. (e.g. WGS84);
- *Projections* - As defined above, it refers to the mathematical equation that was used to project the truly round earth onto a flat surface.

CRS projections

The *"orange peel" analogy* is useful to understand **projections**. If you imagine that the earth is an *orange*, how you *peel it* and then *flatten the peel* is similar to how projections get made.

- A **datum** is the choice of fruit to use. Is the earth an orange, a lemon, a lime, a grapefruit?

Figure 4. Image of citrus.

- A **projection** is how you peel your orange and then flatten the peel.

Figure 5. Image of peeled orange with globe.

Figure 6. Maps of the United States in different projections (Source: opennews.org)

The above image shows maps of the United States in **different projections**. Notice the differences in *shape* associated with each projection. These differences are a direct result of the calculations used to *flatten* the data onto a 2-dimensional map.

- Data from the **same location** but saved in **different projections** will not line up in any GIS software.
- Thus, it's important when working with spatial data to **identify the coordinate reference system** applied to the data **and retain it** throughout data processing and analysis.

21.6 Set up a CRS Projection to UTM coordinates

Let's say that you receive a data frame with coordinates, but *without* a CRS projection. If you want to make a `ggplot` map with it, you know that you can use `st_as_sf()` from the `{sf}` package.

RECAP

As we learned in a previous lesson, for **point** data we need to specify the coordinates and the CRS:

```
fatalities %>%
  st_as_sf(coords = c("x","y"),
           crs = 4326) %>%
  ggplot() +
  geom_sf(alpha = 0.3)
```

However, what if you receive coordinates **different to** *longitude* and *latitude*?

To exemplify this scenario, we are going to use **malaria prevalence in The Gambia**. We use data of malaria prevalence in children obtained at **65 villages** in The Gambia.

21.6.1 Malaria prevalence

First, we download the `gambia.rda` file from internet using its **URL path** with the `{rio}` package.

```
gambia <-
        rio::import("https://github.com/cran/geoR/raw/master/data/gambia.rda")

as_tibble(gambia)
```

```
## # A tibble: 2,035 × 8
##          x       y   pos   age netuse treated green   phc
##      <dbl>   <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl>
##  1 349631. 1458055     1  1783      0       0  40.8     1
##  2 349631. 1458055     0   404      1       0  40.8     1
##  3 349631. 1458055     0   452      1       0  40.8     1
##  4 349631. 1458055     1   566      1       0  40.8     1
##  5 349631. 1458055     0   598      1       0  40.8     1
##  6 349631. 1458055     1   590      1       0  40.8     1
##  7 349631. 1458055     1   589      1       0  40.8     1
##  8 349631. 1458055     0   609      1       0  40.8     1
##  9 349631. 1458055     0   791      1       0  40.8     1
## 10 349631. 1458055     1   829      0       0  40.8     1
## # … with 2,025 more rows
```

It is a data frame with 2035 observations and 8 variables, among them:

- `x`: x coordinate of the village (**UTM**),
- `y`: y coordinate of the village (**UTM**),
- `pos`: presence (1) or absence (0) of malaria in a blood sample taken from the child,

VOCAB

UTM stants for Universal Transverse Mercator, another coordinate

system.

Aggregate data

Data in `gambia` are given at an **individual level**. To get an estimate of the prevalence per village, we need to **aggregate** the malaria tests *by village*.

Unique coordinates

Using the `dplyr::distinct()` function, we can see that `gambia` has 2035 rows and 65 **unique pair** of coordinates. This indicates that 2035 malaria tests were conducted at 65 locations.

```
gambia %>%
  distinct(x,y) %>%
  nrow()
```

```
## [1] 65
```

Use `group_by()` and `summarise()`

Here we use `dplyr` to create a data frame called `gambia_point_summary` containing, **for each village** the:

- *coordinates* (x, y),
- *total* number of tests performed (total),
- number of *positive* tests (positive), and
- malaria *prevalence* (prev).

```
gambia_point_summary <-
  gambia %>%
  group_by(x, y) %>%
  summarize(
    total = n(),
    positive = sum(pos),
    prev = positive / total
  ) %>%
  ungroup()
```

```
gambia_point_summary
```

```
## # A tibble: 65 × 5
##          x       y total positive  prev
```

```
##       <dbl>   <dbl> <int>     <dbl> <dbl>
##  1 349631. 1458055    33        17 0.515
##  2 358543. 1460112    63        19 0.302
##  3 360308. 1460026    17         7 0.412
##  4 363796. 1496919    24         8 0.333
##  5 366400. 1460248    26        10 0.385
##  6 366688. 1463002    18         7 0.389
##  7 368420. 1460569    38        24 0.632
##  8 370400. 1460301    56         7 0.125
##  9 370628. 1499589    57         6 0.105
## 10 374403. 1501392    26         8 0.308
## # … with 55 more rows
```

Set up the CRS projection

Now we can plot the malaria prevalence. However, in order to use `ggplot2` and `geom_sf()`, we need to transform the `data.frame` to an `sf` object.

To use the `st_as_sf()` function from the `{sf}` package, we need to specify a CRS projection. But in this case, the **units** of the `x` and `y` variables are **not** in *Geographic* coordinates (longitude/latitude). Instead, these data coordinates are in **UTM format (Easting/Northing)**, also called **Projected** coordinates.

CRS coordinate systems:

- **Geographic** (or unprojected) reference systems use *longitude and latitude* for referencing a location on the Earth's *three-dimensional* ellipsoid surface.
- **Projected** coordinate reference systems use *easting and northing* Cartesian coordinates for referencing a location on a *two-dimensional* representation of the Earth.

VOCAB

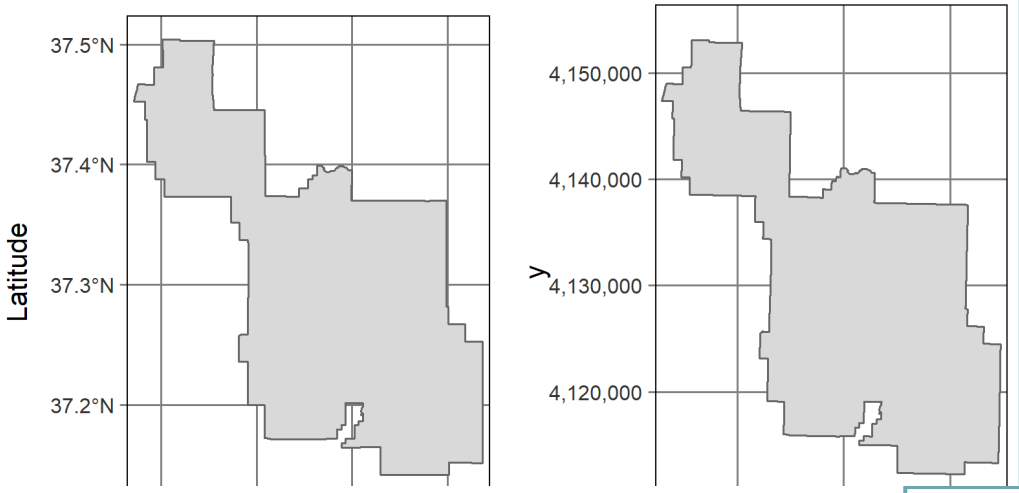


Figure 7. Coordinate systems. Left: Geographic, Right: Projected.

All **Projected CRSs** are *based on* a **Geographic CRS** and rely on *map projections* to convert the three-dimensional surface of the Earth into Easting and Northing (x and y) values in a projected CRS.

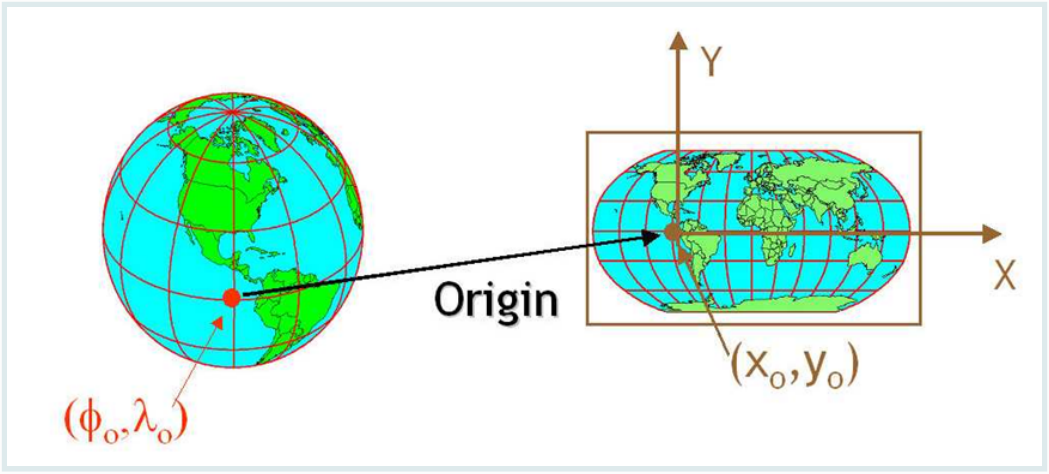


Figure 8. Coordinate systems. Left: Geographic, Right: Projected.

Which of the following options of Coordinate Reference System (CRS) types:

a. `"geographic_crs"`
b. `"projected_crs"`

...corresponds to each of these datasets, given the magnitude of the values in their `x` and `y` columns:

the `parana` dataset?

```
parana <-
        import("https://github.com/cran/geoR/raw/master/data/para

as_tibble(parana$coords)
```

PRACTICE (in RMD) the `fatalities` dataset?

```
pacman::p_load(cholera)
as_tibble(fatalities)
```

Set UTM Projected coordinates with `st_as_sf()`

First, we need to set **UTM coordinates**. For this, we specify the projection of The Gambia, that is, **UTM zone 28** (`"+proj=utm +zone=28"`) in the `st_as_sf()` function of the `{sf}` package.

```
gambia_projected <- gambia_point_summary %>%
  # first, specify the projection of gambia
  # UTM zone 28
  st_as_sf(coords = c("x", "y"),
           crs = "+proj=utm +zone=28")

gambia_projected
```

```
## Simple feature collection with 65 features and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 349631.3 ymin: 1458055 xmax: 622086.1 ymax: 1510610
## CRS:           +proj=utm +zone=28
## # A tibble: 65 × 4
##    total positive  prev           geometry
##  * <int>    <dbl> <dbl>        <POINT [m]>
##  1    33       17 0.515 (349631.3 1458055)
##  2    63       19 0.302 (358543.1 1460112)
##  3    17        7 0.412 (360308.1 1460026)
##  4    24        8 0.333 (363795.7 1496919)
##  5    26       10 0.385 (366400.5 1460248)
##  6    18        7 0.389 (366687.5 1463002)
##  7    38       24 0.632 (368420.5 1460569)
##  8    56        7 0.125 (370399.5 1460301)
##  9    57        6 0.105 (370627.7 1499589)
## 10    26        8 0.308 (374402.6 1501392)
## # … with 55 more rows
```

Confirm the presence of the:

- CRS text (`CRS: +proj=utm +zone=28`) inside the **header** of the new `sf` object, and
- the **unit** the `geometry` column in **meters** (`<POINT [m]>`).

VOCAB

- The **UTM** system divides the Earth into **60 zones** of *6 degrees of longitude in width*. Each of the zones uses a transverse Mercator

projection that maps a region of large north-south extent.

- To **get** the *UTM zones* of various parts of the world, you could use online interactive maps, or gridded images available in wikipedia.

In the **UTM** system, a **position** on the Earth is given by the:

- UTM *zone* number,
- *Hemisphere* (north or south), and
- Easting and northing *coordinates* in the zone which are measured in **meters**.
  - *Eastings* are referenced from the central meridian of each zone, and
  - *northings* are referenced from the equator.

`parana_data` contains the average rainfall over different years for the period May-June (dry-season). It was collected at 143 recording stations throughout *Parana State, Brasil*.

Set UTM coordinate system to the `parana_data`.

```
parana_data <- as_tibble(parana$coords) %>%
  mutate(Rainfall = parana$data)
```

Transform to Geographic coordinates with `st_transform()`

We can transform the UTM *projected* coordinates to *geographic* coordinates (**longitude/latitude** with datum **WGS84**) using `st_transform()` where we set CRS to `"+proj=longlat +datum=WGS84"`.

```
gambia_geographic <- gambia_projected %>%
  # second, transform
  # projected coordinates to
  # geographic coordinates
  st_transform(crs = "+proj=longlat +datum=WGS84")

gambia_geographic
```

```
## Simple feature collection with 65 features and 3 fields
## Geometry type: POINT
```

```
## Dimension:     XY
## Bounding box:  xmin: -16.38755 ymin: 13.18541 xmax: -13.87273 ymax:
13.66438
## CRS:           +proj=longlat +datum=WGS84
## # A tibble: 65 × 4
##    total positive  prev             geometry
##  * <int>    <dbl> <dbl>          <POINT [°]>
##  1    33       17 0.515 (-16.38755 13.18541)
##  2    63       19 0.302 (-16.30543 13.20444)
##  3    17        7 0.412 (-16.28914 13.20374)
##  4    24        8 0.333 (-16.25869 13.53742)
##  5    26       10 0.385 (-16.23294 13.20603)
##  6    18        7 0.389 (-16.23041 13.23094)
##  7    38       24 0.632 (-16.21431 13.20902)
##  8    56        7 0.125 (-16.19604 13.20668)
##  9    57        6 0.105 (-16.19569 13.56187)
## 10    26        8 0.308 (-16.16088 13.57834)
## # … with 55 more rows
```

Confirm the **update** of the:

- CRS text to `CRS: +proj=longlat +datum=WGS84` inside the **header**, and
- the **units** of the `geometry` column to **degrees** (`<POINT [°]>`).

A **PROJ string** includes the following information:

- `+proj=`: the projection of the data (e.g. `utm`, `longlat`, or `laea`)
- `+zone=`: the zone of the data, specific to the UTM projection (e.g. `28`)
- `+datum=`: the datum use (e.g. `WGS84`)
- `+units=`: the units for the coordinates of the data (e.g. `m`)

With the UTM coordinate system data stored in `q6`:

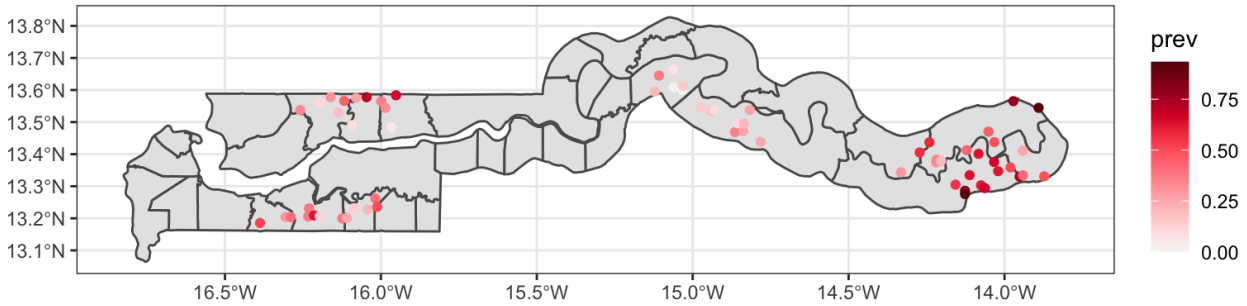
Transform its Projected CRS to a Geographic CRS using the `longitude/latitude` projection with datum `WGS84`.

Map prevalences

Now that you have set up the right CRS projection to your data, you can **overlap** these points with other Vector data objects:

```
gambia_adm_2 <- geoboundaries(country = "Gambia", adm_lvl = 2)
```

```
ggplot() +
  geom_sf(data = gambia_adm_2) +
  geom_sf(data = gambia_geographic, mapping = aes(color = prev)) +
  colorspace::scale_color_continuous_sequential(palette="Reds 3")
```



Which CRS to use?

"There exist no all-purpose projections, all involve distortion when far from the center of the specified frame" (Bivand, Pebesma, and Gómez-Rubio 2013).

- When **Geographic CRS**, the answer is often **WGS84**.
  - It is used by default for web mapping, in *GPS* datasets, and vector datasets.
  - WGS84 is the most common CRS in the world, so it is worth knowing its *EPSG code: 4326*.
  - This 'magic number' can be used to convert objects with unusual projected CRSs into something that is widely

understood.
- About **Projected CRS**,
  - It is often a choice made by a *public mapping agency*.
  - With local data sources, work with the CRS in which the data was provided, to ensure *compatibility*, even if the official CRS is not the most accurate.

21.7 Wrap up

In this lesson, we have learned how to **manage** a CRS *projection* in `ggplot` maps and `sf` objects, how projections are **codified** with *EPSG* codes and *PROJ* strings, and how to **transform** the CRS between different *coordinate systems* (projected and geographic).

In the next lesson, we are going to use all our previous learning to built one single thematic map by **layers**, and enrich them with **text** and **labels** referring to specific places or regions, and important map elements like **scale bars** and a **north arrow**!

Contributors

The following team members contributed to this lesson:

ANDREE VALLE CAMPOS

R Developer and Instructor, The Graph Network
Motivated by reproducible science and education.

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- *Moreno, M., Basille, M. Drawing beautiful maps programmatically with R, sf and ggplot2 – Part 1: Basics*. (2018). Retrieved 10 May 2022, from https://r-spatial.org/r

/2018/10/25/ggplot2-sf.html

- *Data carpentry. Introduction to Geospatial Concepts: Coordinate Reference Systems*. (2021). Retrieved 15 May 2022, from https://datacarpentry.org/organization-geospatial/03-crs/index.html

- *Moraga, Paula. Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny. Chapter 9: Spatial modeling of geostatistical data. Malaria in The Gambia*. (2019). Retrieved 10 May 2022, from https://www.paulamoraga.com/book-geospatial/sec-geostatisticaldataexamplespatial.html

- *Carrasco-Escobar, G., Barja, A., Quispe, J. [Visualization and Analysis of Spatial Data in Public Health]*. (2021). Retrieved 15 May 2022, from https://www.reconlearn.org/post/spatial-analysis-1-spanish.html

22

Geospatial analysis: additional layers

22.1 Learning objectives

1. Add **external** data coordinates within the scripts using the `tribble()` function.
2. Add text and **labels** to ggplot maps using the `{ggsflabel}` package.
3. Add color **palettes** to spatial data using the `{colorspace}` package.
4. Add arrow and scale **annotations** in ggplot maps using the `{ggspatial}` package.

22.2 Prerequisites

This lesson requires the following packages:

```
if(!require('pacman')) install.packages('pacman')

pacman::p_load(tidyverse,
               colorspace,
               ggspatial,
               janitor,
               ggplot2,
               spData,
               units,
               sf)

pacman::p_load_gh("yutannihilation/ggsflabel",
                  "afrimapr/afrihealthsites",
                  "afrimapr/afrilearndata",
                  "wmgeolab/rgeoboundaries")
```

22.3 Introduction

Until now, we have learnt general concepts about geospatial visualization in independent lessons.

The **modular** approach of `ggplot2` allows to successively add all of them in different **layers**. For instance, study sites or administrative delineations, and a visual representation of their characteristics in a single map, using useful **color palettes**.

These enriched thematic maps also require to contain **text** and **labels** referring to specific places or regions, and important map elements like **scale bars** and a **north arrow**, as will be illustrated in this part.

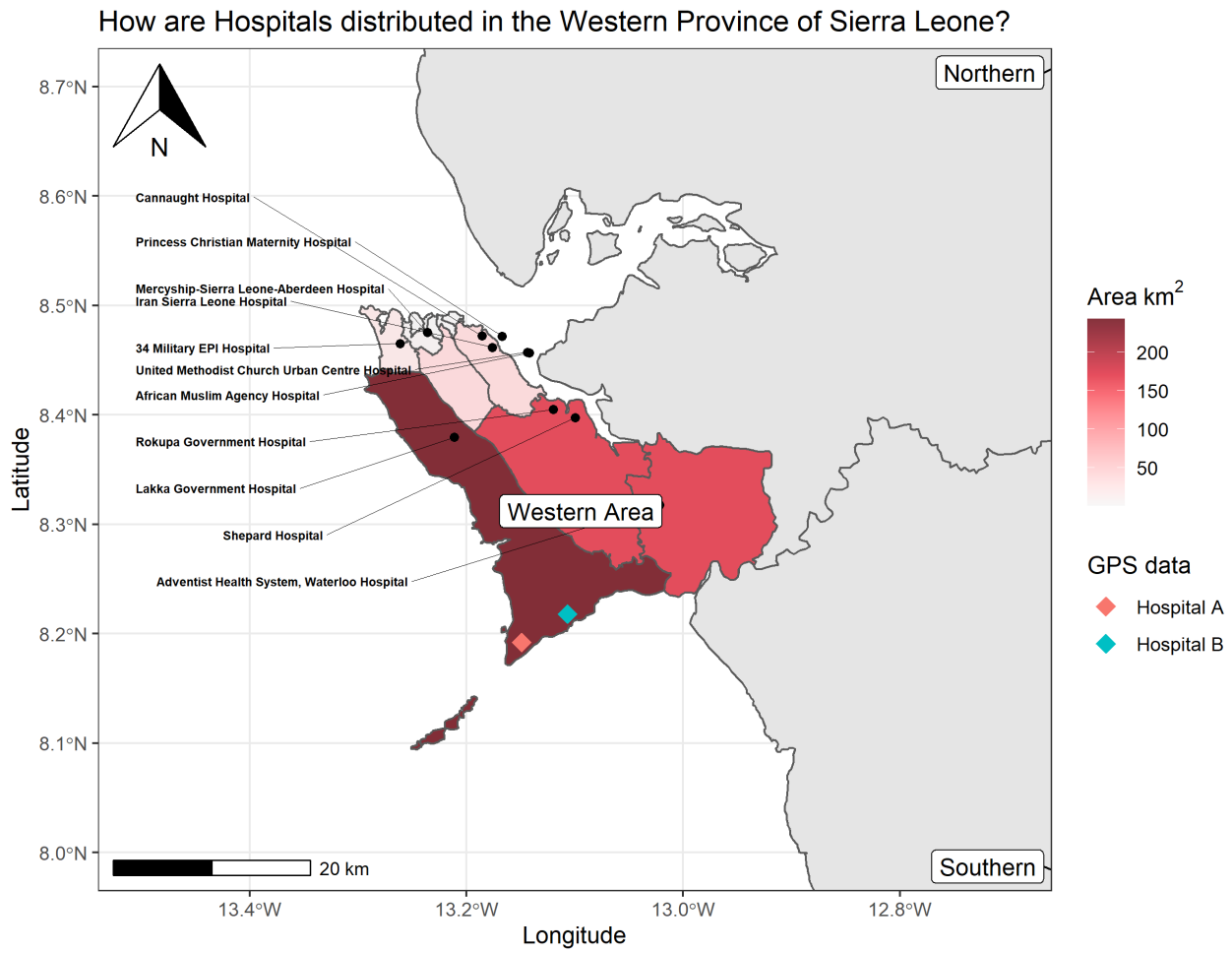


Figure 1. Ggplot map with multiple layers.

22.4 Build an informative map

We will create a map to figure out how are Hospitals distributed in the Western Province of Sierra Leone. For this, we are going to simulate the location of two Hospital manually collected in the field, and also retrieve real Hospital information from a public repository.

Add field sites with `tribble()`

We start by defining two Hospital sites (point data), according to their longitude and latitude, collected by GPS devices in the field, using the `tribble()` function from the `{tibble}` package:

```
sites <- tribble(~gps_name, ~gps_longitude, ~gps_latitude,
                 "site A", -13.1491617, 8.1920813,
                 "site B", -13.1066807, 8.2180983
                 )

sites
```

```
## # A tibble: 2 × 3
##   gps_name gps_longitude gps_latitude
##   <chr>            <dbl>        <dbl>
## 1 site A           -13.1         8.19
## 2 site B           -13.1         8.22
```

Create a tibble object with three columns (`gps_name`, `gps_longitude`, `gps_latitude`) with the GPS location of one `"household"` with longitude `-13.1856942` and latitude `8.2851963`. Use the `tribble()` function.

`tribble` is better used for a minimum amount of observations. For **larger amounts** you may prefer to read your data from a *csv* or *excel* file.

These sites belong to Sierra Leone:

```
sierra_leone <- geoboundaries(country = "Sierra Leone", adm_lvl = 1)
```

To add these geographic coordinates to a country map, we need to use the power of `sf`.

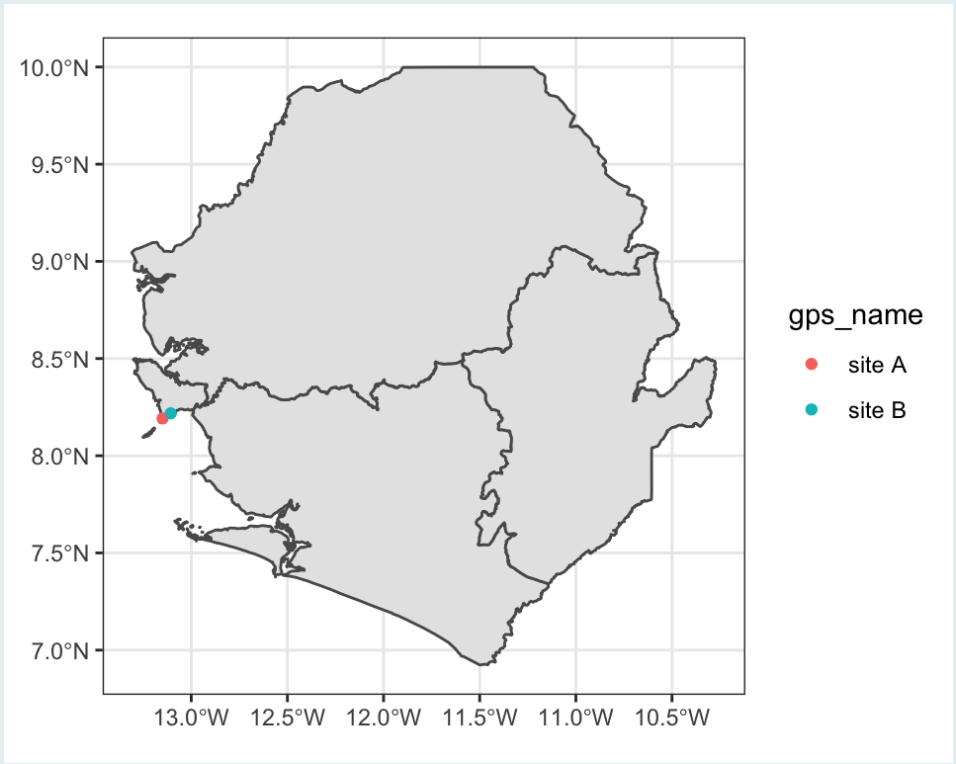
Converting the data frame to a `sf` object allows to rely on `sf` to handle on the fly the **coordinate system** (both *projection* and *extent*), which can be very useful if the two objects (here `sierra_leone`, and `sites`) are not in the same projection.

To achieve this, the **projection** (here WGS84, which is the CRS code `#4326`) has to be a priori defined in the `sf` object with the `st_as_sf()` function (*for more information: see previous lessons*) :

```
sites_sf <- sites %>%
  st_as_sf(coords = c("gps_longitude","gps_latitude"),
           crs = 4326)
```

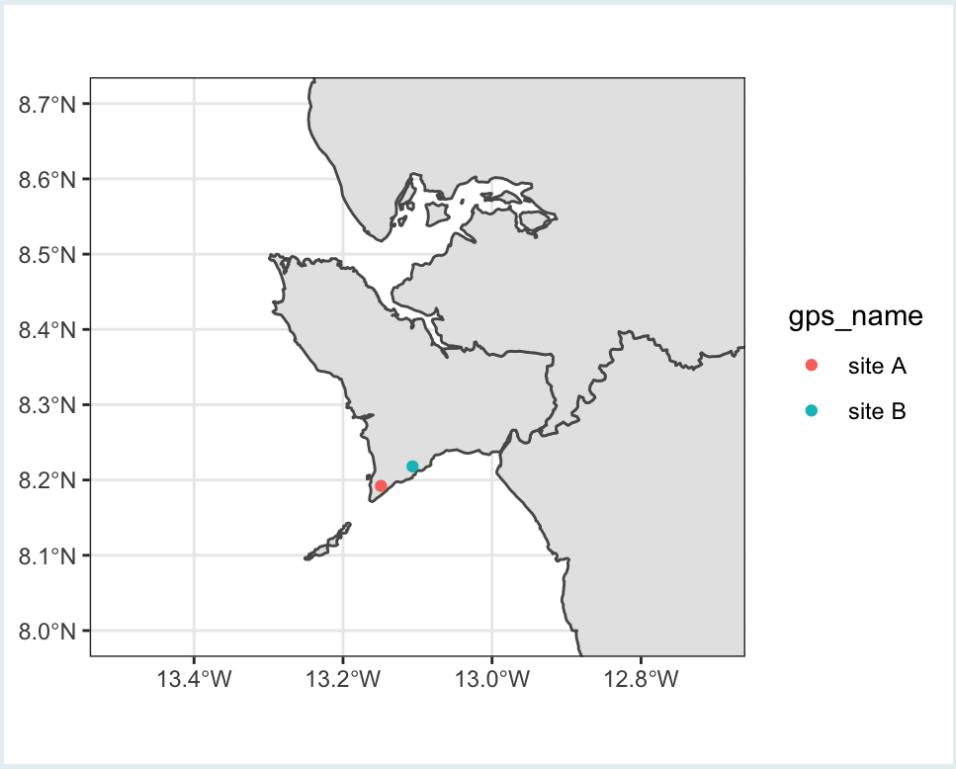
Now we can make a map with `sierra_leone` and `sites_sf` objects, and add the type of data (`gps_name`) in the legend:

```
ggplot() +
  geom_sf(data = sierra_leone) +
  geom_sf(data = sites_sf, mapping = aes(color = gps_name))
```



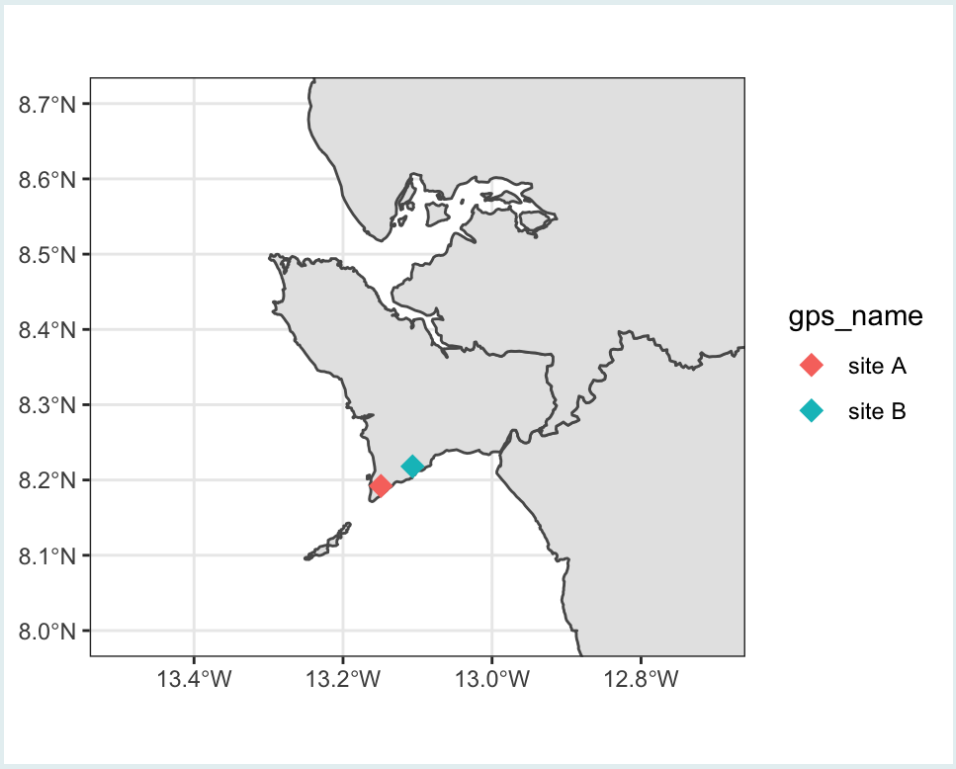
We can use `coord_sf()` after all `geom_sf()` calls, to **zoom in** to our area of interest inside Sierra Leone:

```
ggplot() +
  # geometries
  geom_sf(data = sierra_leone) +
  geom_sf(data = sites_sf, mapping = aes(color = gps_name)) +
  # map extent
  coord_sf(xlim = c(-13.5,-12.7), ylim = c(8.0,8.7))
```



As such, we can adjust all characteristics of **points** (e.g. color of the outline and the filling, shape, size, etc.), for each `geom_sf()` layer. In this example, we set the two points as filled diamonds (`shape = 18`) with a bigger size (`size = 2`):

```
ggplot() +
  # geometries
  geom_sf(data = sierra_leone) +
  geom_sf(data = sites_sf, mapping = aes(color = gps_name),
          shape = 18, size = 4) +
  # map extent
  coord_sf(xlim = c(-13.5,-12.7), ylim = c(8.0,8.7))
```

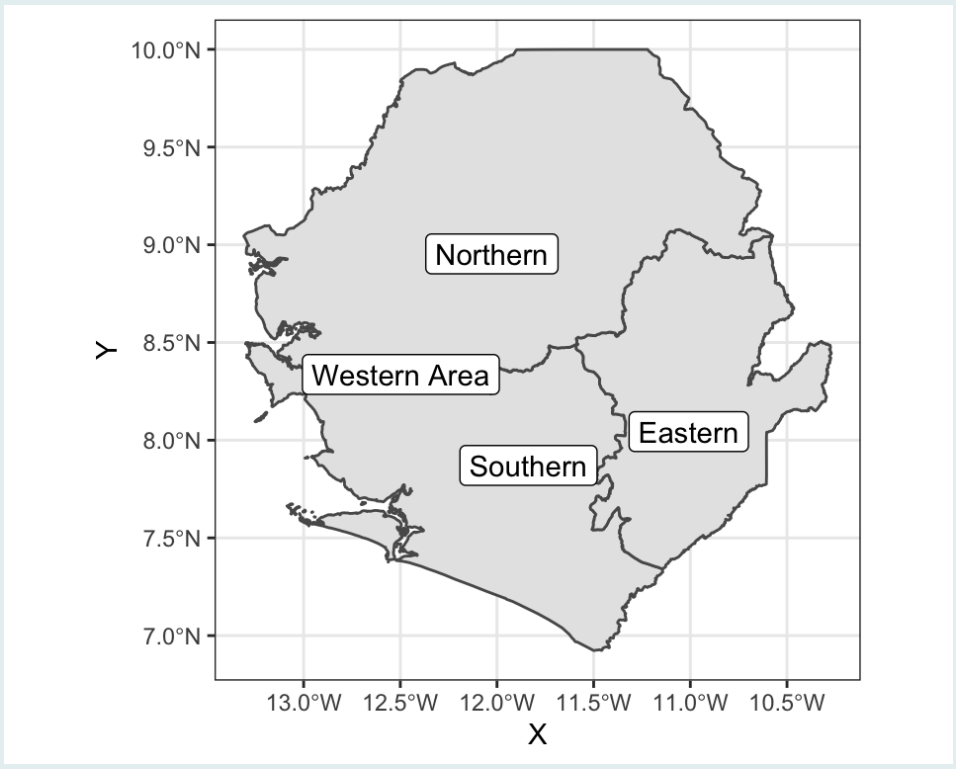


22.4.1 Add province labels with `geom_sf_label_repel()`

It would be informative to add finer administrative information on top of the previous map, starting with administrative borders (`sf` object: polygon data) and their names. The package `ggsflabel` provides functions to add labels to `sf` objects (points and polygons) in maps.

Province names are part of this data, as the `shapeName` variable.

```
sierra_leone %>%
  ggplot() +
  geom_sf() +
  geom_sf_label_repel(aes(label=shapeName))
```



PRACTICE (in RMD)

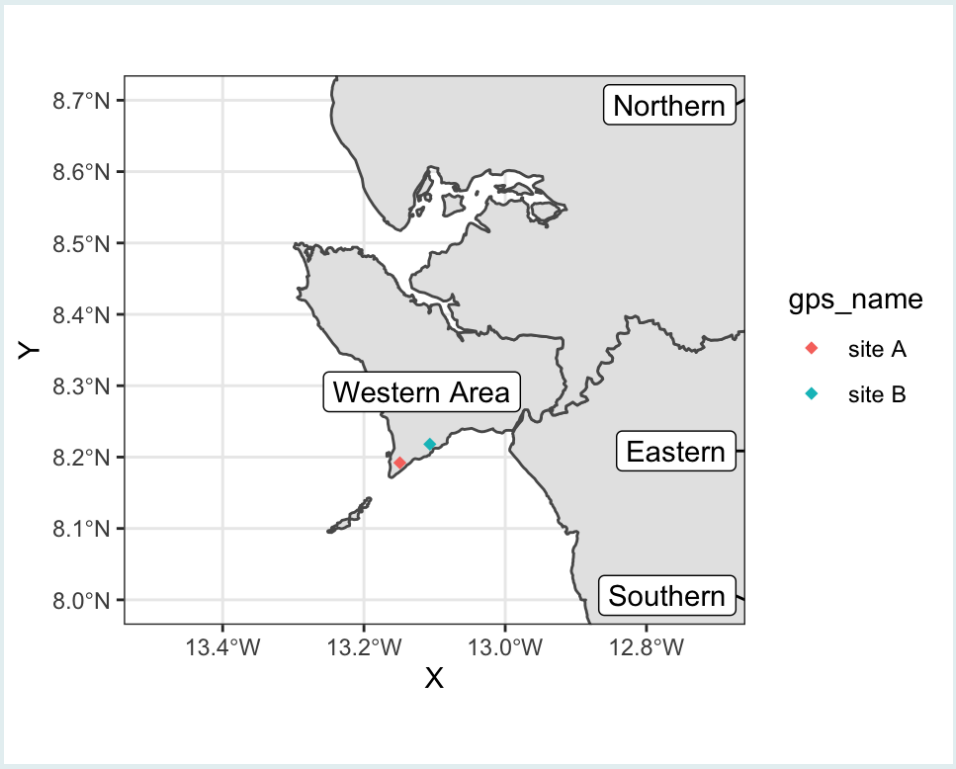
The `zimbabwe_adm1` object contains the boundaries of all the provinces in Zimbabwe.

```
zimbabwe_adm1 <- geoboundaries(country = "Zimbabwe", adm_lvl =
        1)
```

Create a map of Zimbabwe with labels for the name of each of its provinces.

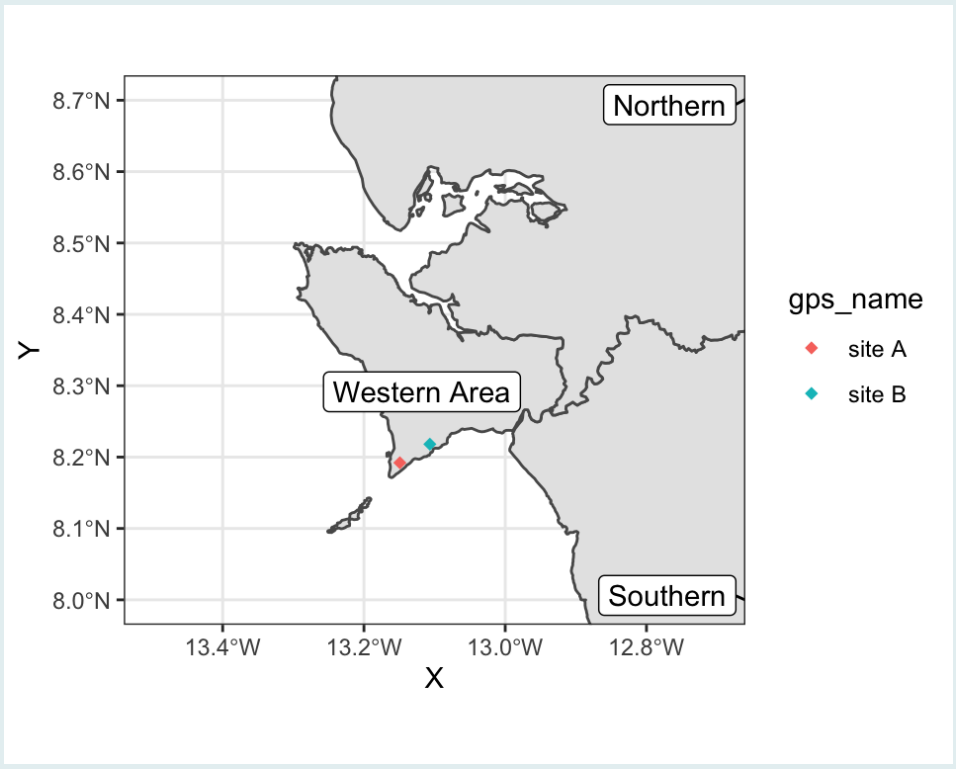
To continue building the complex map introduced at the beginning of the lesson, province data is directly plotted as an additional `sf` layer using `geom_sf()`. In addition, province names will be added using `geom_sf_label_repel()`, as well as the label (from `shapeName`), and a relatively big font size.

```
ggplot() +
  # geometries
  geom_sf(data = sierra_leone) +
  geom_sf(data = sites_sf, mapping = aes(color = gps_name),
          shape = 18, size = 2) +
  # labels
  geom_sf_label_repel(data = sierra_leone, mapping = aes(label=shapeName)) +
  # map extent
  coord_sf(xlim = c(-13.5,-12.7), ylim = c(8.0,8.7))
```



We can drop the province name of "Eastern". For this, we can use `filter()` from the `{dplyr}` package:

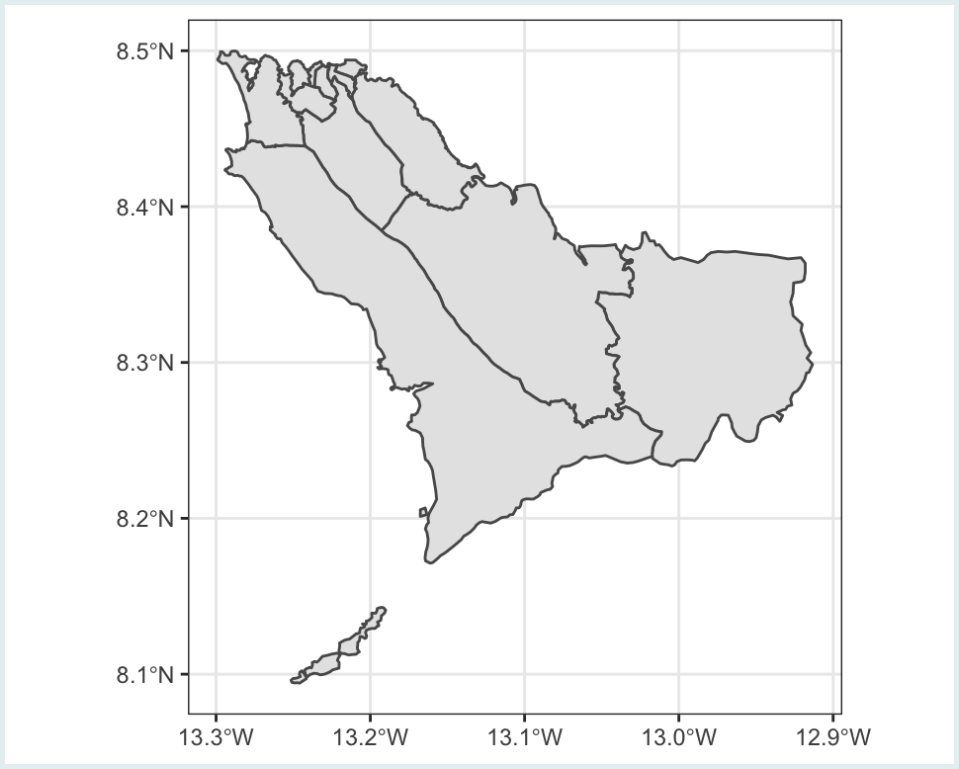
```
ggplot() +
  # geometries
  geom_sf(data = sierra_leone) +
  geom_sf(data = sites_sf, mapping = aes(color = gps_name),
          shape = 18, size = 2) +
  # label
  geom_sf_label_repel(data = sierra_leone %>% filter(shapeName!="Eastern"),
                      mapping = aes(label=shapeName)) +
  # map extent
  coord_sf(xlim = c(-13.5,-12.7), ylim = c(8.0,8.7))
```



Fill district data with {colorspace}

Districts (polygon data) can be retrieved from local **shapefile** data. This time, only districts from Western province are retained:

```
sierra_leone_west <-
  sf::read_sf(dsn = here::here("ch06_basic_geospatial_viz",
                               "data","gis","shp",
                               "sle_adm3.shp"),
              quiet = TRUE) %>%
  filter(admin1Name=="Western")

ggplot(data = sierra_leone_west) +
  geom_sf()
```



This time, for all the districts from the province retained, we compute their area using `st_area()` from the package `sf`:

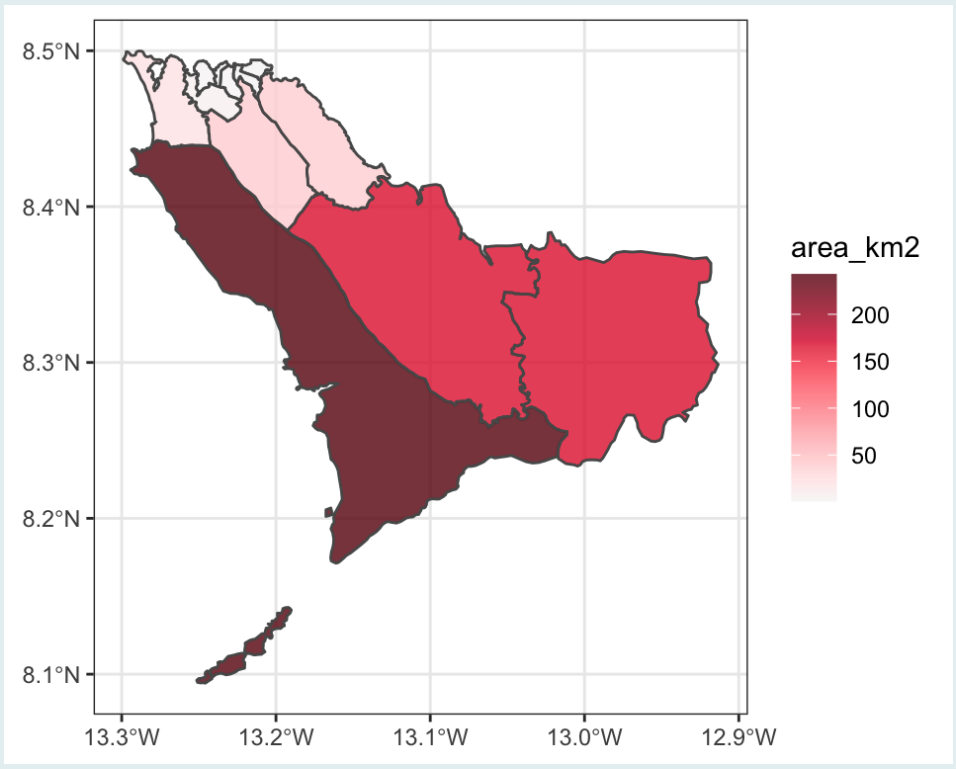
```
sierra_leone_shp <- sierra_leone_west %>%
  # calculate area of each polygon
  mutate(area_m2 = st_area(.)) %>%
  # convert m2 to km2
  mutate(area_km2 = units::set_units(.$area_m2,km^2)) %>%
  # transform the variable to numeric
  mutate(area_km2 = as.numeric(area_km2))

sierra_leone_shp %>%
  select(area_km2)
```

```
## Simple feature collection with 12 features and 1 field
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -13.29894 ymin: 8.094272 xmax: -12.91333 ymax:
8.499809
## Geodetic CRS:  WGS 84
## # A tibble: 12 × 2
##    area_km2
geometry
##       <dbl>
<MULTIPOLYGON [°]>
##  1   167.    (((-13.02082 8.381989, -13.0188 8.377952, -13.01781 8.377952,
-13.01603 8.377952, -1...
##  2    38.9   (((-13.21496 8.474341, -13.21479 8.474289, -13.21465 8.474296,
-13.21455 8.474298, -...
```

```
##  3  167.    (((-13.12215 8.413156, -13.12155 8.413105, -13.12013 8.413218,
-13.11941 8.413382, -...
##  4  243.    (((-13.24441 8.095223, -13.24574 8.094272, -13.24742 8.094518,
-13.2491 8.094406, -1...
##  5    2.30  (((-13.22646 8.489716, -13.22648 8.48955, -13.22644 8.489513,
-13.22663 8.489229, -1...
##  6    1.75  (((-13.2129 8.494033, -13.21076 8.494026, -13.21013 8.494041,
-13.2096 8.494025, -13...
##  7    1.83  (((-13.22653 8.491883, -13.22647 8.491853, -13.22642 8.49186,
-13.22633 8.491814, -1...
##  8    0.796 (((-13.23154 8.491768, -13.23141 8.491566, -13.23144 8.49146,
-13.23131 8.491294, -1...
##  9   20.8   (((-13.28529 8.497354, -13.28456 8.496497, -13.28403 8.49621,
-13.28338 8.496086, -1...
## 10    2.23  (((-13.24677 8.493453, -13.24669 8.493285, -13.2464 8.493132,
-13.24627 8.493131, -1...
## 11    6.69  (((-13.25698 8.485518, -13.25685 8.485501, -13.25668 8.485505,
-13.25657 8.485504, -...
## 12   37.9   (((-13.20465 8.485758, -13.20461 8.485698, -13.20449 8.485757,
-13.20431 8.485577, -...
```

We can now fill in the districts using their **area** to visually identify the largest counties. For this, we use the `{colorspace}` package with some transparency. In this case, `area_km2` is a *continuous* variable, with a *sequential* scale type:

```
sierra_leone_shp %>%
  ggplot() +
  # geometry
  geom_sf(aes(fill = area_km2)) +
  # aesthetic
  colorspace::scale_fill_continuous_sequential(palette="Reds 3", alpha = 0.8)
```



The **scales** are called via the scheme

```
scale_<aesthetic>_<datatype>_<colorscale>()
```

VOCAB

where:

- `<aesthetic>` is the name of the *aesthetic* (fill, color, colour).
- `<datatype>` is the type of the *variable* plotted (discrete, continuous, binned).
- `<colorscale>` sets the type of the color *scale* used (qualitative, sequential, diverging, divergingx).

PRACTICE (in RMD)

This is a ggplot map with the `afriairports` object:

```
afriairports %>%
  ggplot() +
  geom_sf(aes(color = elevation_ft))
```

Paste this code to your answer and:

Update the *color* aesthetic of the *continuous* variable `elevation_ft` of this map to a *diverging* scale, with a midpoint (`mid`) in `5000`.

A **Sequential** colors scale indicate:

- Which values are *larger* or *smaller than* which other ones, and
- How *distant two specific values are from each other*. This implies that the color scale needs to be perceived to vary uniformly across its entire range.

A **Diverging** color scale allows to:

- visualize the *deviation of data values* in one of two directions relative to a *neutral midpoint*, usually is represented by a light color,
- For example, a dataset containing both *positive* and *negative* numbers, and show how far in either direction it *deviates from zero*.

You can use `{colorspace}` package to:

- emulate color vision deficiency heatmaps with palettes, and
- access colorblind-friendly palettes like Viridis and others.

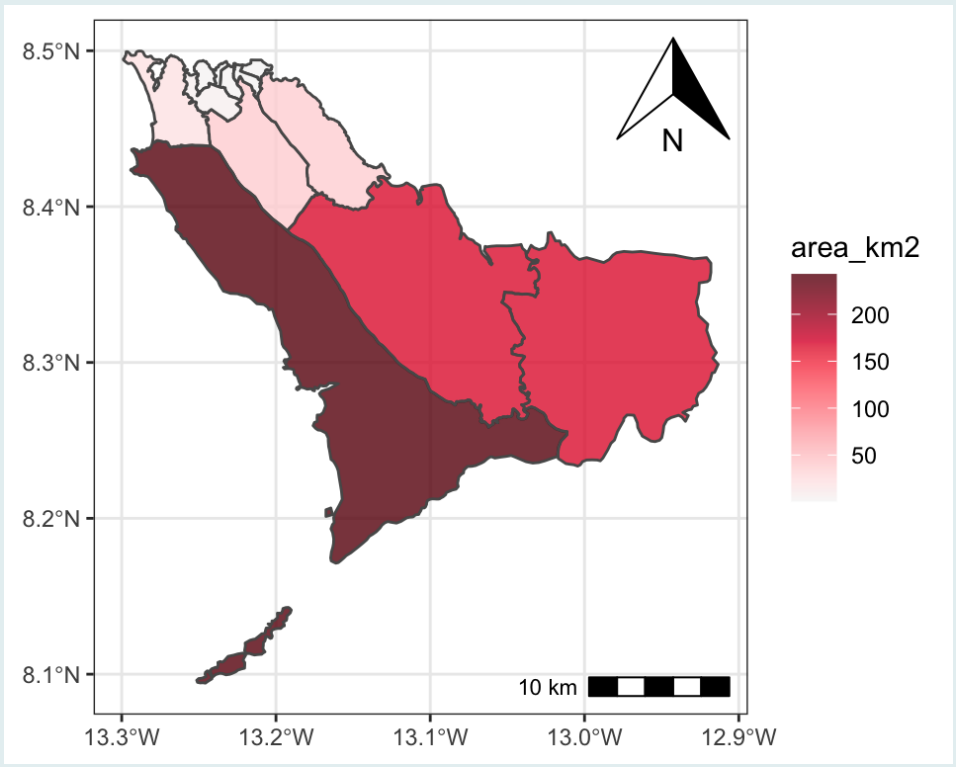
Add a Scale bar and North arrow with `{ggspatial}`

We introduce here the package `{ggspatial}`, which provides easy-to-use functions to create a scale bar and north arrow on a `ggplot` map:

- `annotation_north_arrow()` allows to add the north symbol and
- `annotation_scale()` a scale bar.

```
sierra_leone_shp %>%
  ggplot() +
  # geometry
  geom_sf(aes(fill = area_km2)) +
  # aesthetic
```

```
colorspace::scale_fill_continuous_sequential(palette="Reds 3", alpha = 0.8)
      +
annotation_north_arrow(location="tr") +
annotation_scale(location="br")
```



The **location** of the scale bar and north arrow are by default in the bottom left (`"bl"`) side of the map. They can be specified using the `location` argument with `"tr"` for top right, `"bl"` for bottom left, etc.

PRACTICE (in RMD)

This is a ggplot map with the `zimbabwe_adm1` object:

```
zimbabwe_adm1 %>%
  ggplot() +
  geom_sf()
```

Paste this code to your answer and:

Add a Scale bar located in the `bottom right` of the map, and a North arrow in the `top left`.

SIDE NOTE

- The North arrow style in `annotation_north_arrow()` can also be

SIDE NOTE

adjusted using the `style` argument.

- Note that scale distance is set to `"km"` by default in `annotation_scale()`; you can set it in "m", "cm", "mi", "ft", or "in".

Add Health site names with `geom_sf_text_repel()`

To make a more complete map of the Western province of Sierra Leona, Health facilities (`sf` object: point data) will be added to the map.

Instead of looking up coordinates manually, the package `{afrihealthsites}` provides a function `afrihealthsites()`, which allows to retrieve geographic coordinates of African health facilities from different sources:

You can run the following code to retrieve geographic coordinates of all the health facilities in Sierra Leone available in the `who` database:

```
sle_healthsites_all <- afrihealthsites(country = "Sierra Leone",
                                       datasource='who',
                                       plot = FALSE,
                                       returnclass = "dataframe") %>%
  janitor::clean_names()

sle_healthsites_all
```

```
## # A tibble: 1,120 × 12
##    country      admin1  facility_name
facility_type ownership   lat  long ll_source iso3c facility_type_9  tier
tier_name
##    <chr>        <chr>   <chr>                                    <chr>
<chr>     <dbl> <dbl> <chr>     <chr> <chr>           <dbl> <chr>
##  1 Sierra Leone Eastern Ahmadiyya Mission Hospital             Mission
Hosp… FBO         7.89 -11.2 GPS       SLE   Hospital            3 Tier3 pr…
##  2 Sierra Leone Eastern Baama Community Health Centre          Community
He… MoH         8.36 -11.3 GPS       SLE   Community Heal…     2 Tier2 he…
##  3 Sierra Leone Eastern Baiama Community Health Post           Community
He… MoH         8.52 -11.0 GPS       SLE   Community Heal…     1 Tier1 he…
##  4 Sierra Leone Eastern Baiima Community Health Post           Community
He… MoH         8.03 -10.8 GPS       SLE   Community Heal…     1 Tier1 he…
##  5 Sierra Leone Eastern Baiwala Community Health Post          Community
He… MoH         8.00 -10.6 GPS       SLE   Community Heal…     1 Tier1 he…
##  6 Sierra Leone Eastern Bambara Kaima Community Health Post    Community
He… MoH         8.48 -11.3 GPS       SLE   Community Heal…     1 Tier1 he…
##  7 Sierra Leone Eastern Bambara Maternal and Child Health Po… Maternal &
C… MoH         8.32 -11.3 GPS       SLE   Health Post         1 Tier1 he…
##  8 Sierra Leone Eastern Bambawulo Community Health Post        Community
He… MoH         8.01 -11.1 GPS       SLE   Community Heal…     1 Tier1 he…
##  9 Sierra Leone Eastern Bandajuma Community Health Centre      Community
He… MoH         8.31 -10.8 GPS       SLE   Community Heal…     2 Tier2 he…
## 10 Sierra Leone Eastern Bandajuma Kpolihun Community Health … Community
```

```
He… MoH         8.02 -10.9 GPS       SLE    Community Heal…      1 Tier1 he…
## # … with 1,110 more rows
```

Access to all the health facilities of `Zimbabwe` from the `healthsites` data source using the `afrihealthsites()` function.

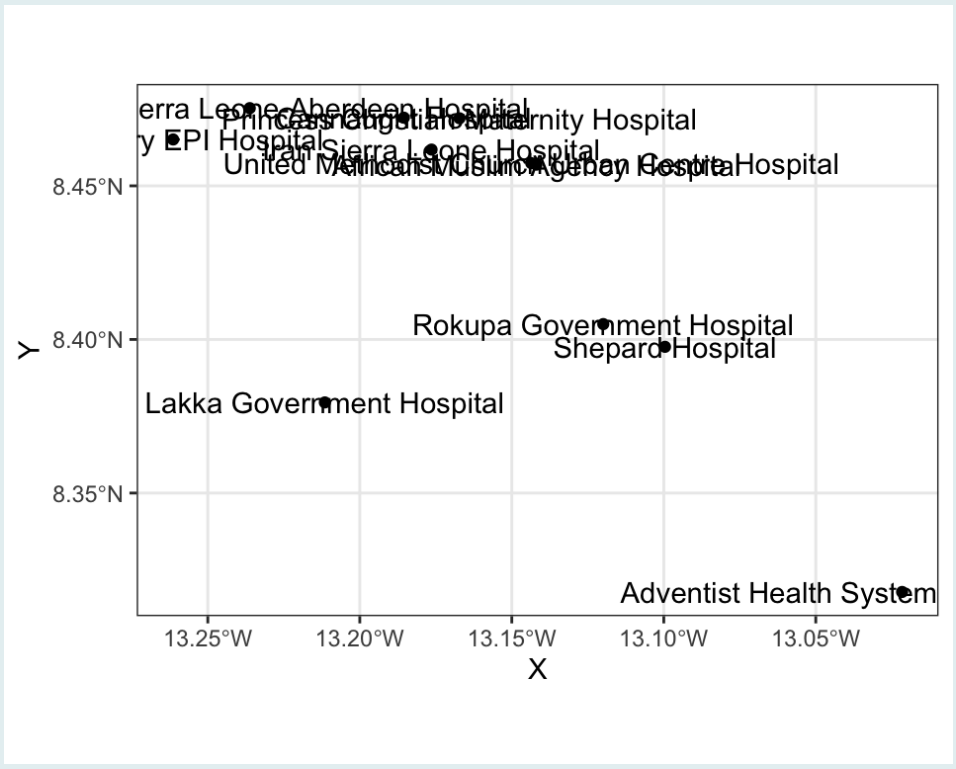
According to the three-tier health delivery classification, the highest tier (Tier 3) includes county hospitals in rural regions and city hospitals in urban regions, which are responsible for most inpatient services as well as teaching and research missions (Wang, 2021).}

We can now keep the Tier 3 health facilities inside the Western province, and convert the data frame with coordinates to `sf` format:

```
sle_healthsites_set <- sle_healthsites_all %>%
  filter(admin1=="Western Area") %>%
  filter(tier==3) %>%
  st_as_sf(coords = c("long","lat"),
           crs = 4326)
```

We add hospital locations and names as text on the map with `geom_sf_text()`:

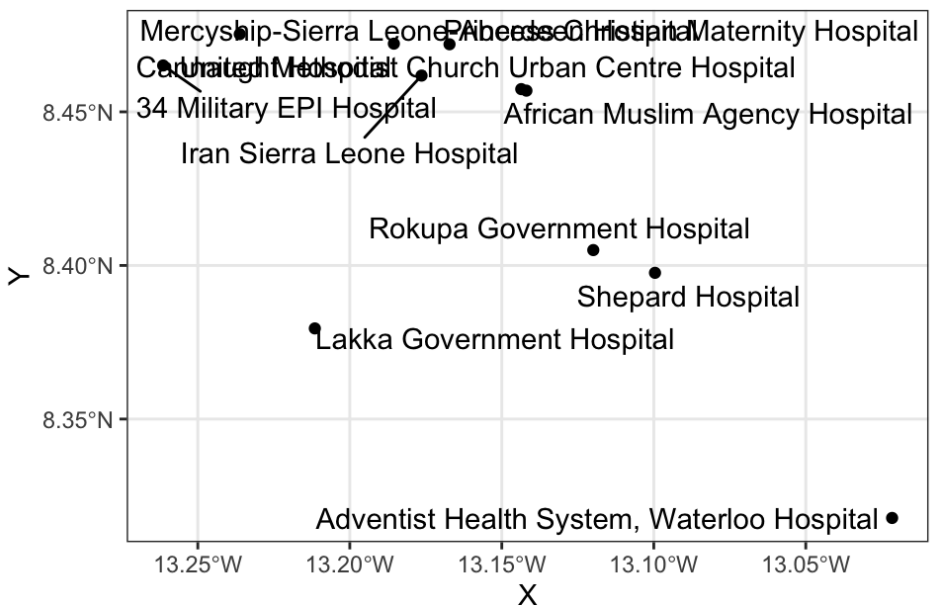
```
sle_healthsites_set %>%
  ggplot() +
  # geometry
  geom_sf() +
  # label
  geom_sf_text(mapping = aes(label= facility_name))
```



This is not really satisfactory, as the names overlap on the points, and they are not easy to read on the grey background. The package `{ggsflabel}` offers a very flexible approach (inspired in `{ggrepel}`) to deal with label placement in `ggplot` maps with `sf` objects (with `geom_sf_text_repel` and `geom_sf_label_repel`), including automated movement of labels in case of overlap.

We use it here to "nudge" the labels away, and connect them to the city locations:

```
sle_healthsites_set %>%
  ggplot() +
  # geometry
  geom_sf() +
  # label
  geom_sf_text_repel(mapping = aes(label= facility_name))
```



This is a ggplot map of all the `hospital` facilities of `Zimbabwe`:

```
afrihealthsites(country = 'Zimbabwe',
                        datasource='healthsites',
                        plot = FALSE,
                        returnclass = 'dataframe') %>%
  filter(amenity == 'hospital') %>%
  ggplot() +
  geom_sf()
```

Paste this code to your answer and:

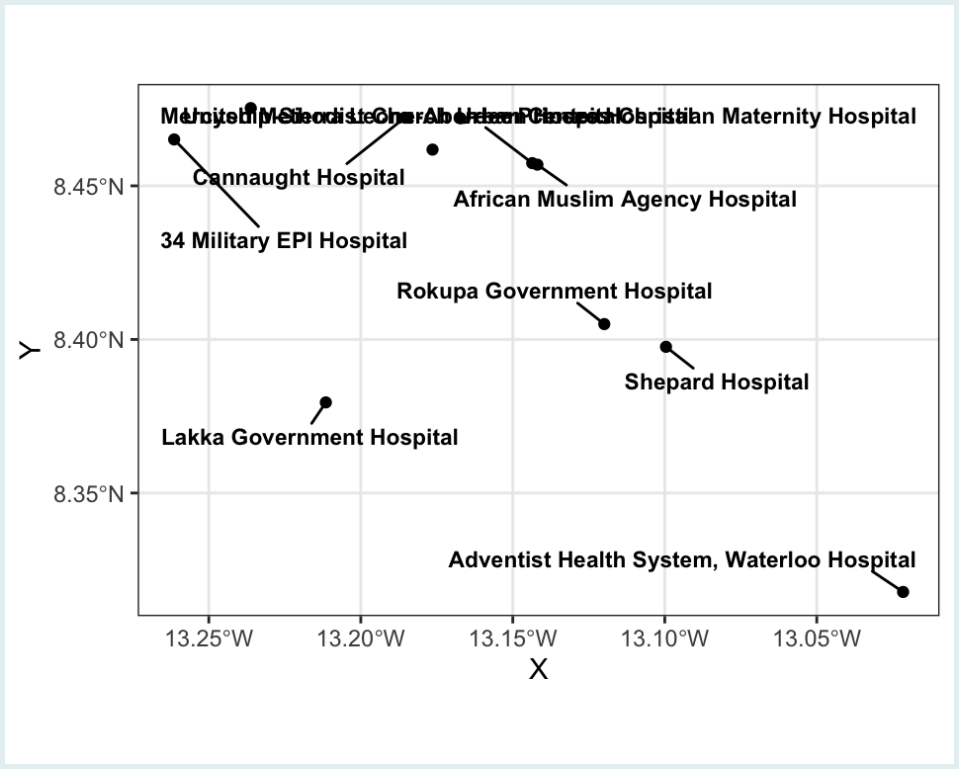
Add their names as text without overlaps to a `ggplot` map.

Additionally, we can manually set `{ggrepel}` arguments to improve its output:

- The size (argument `size`);
- The type of font of the text (`fontface`);
- The force of repulsion between overlapping text labels (`force`);
- The additional padding around the each text label (`box.padding`).

```
sle_healthsites_set %>%
  ggplot() +
  # geometry
```

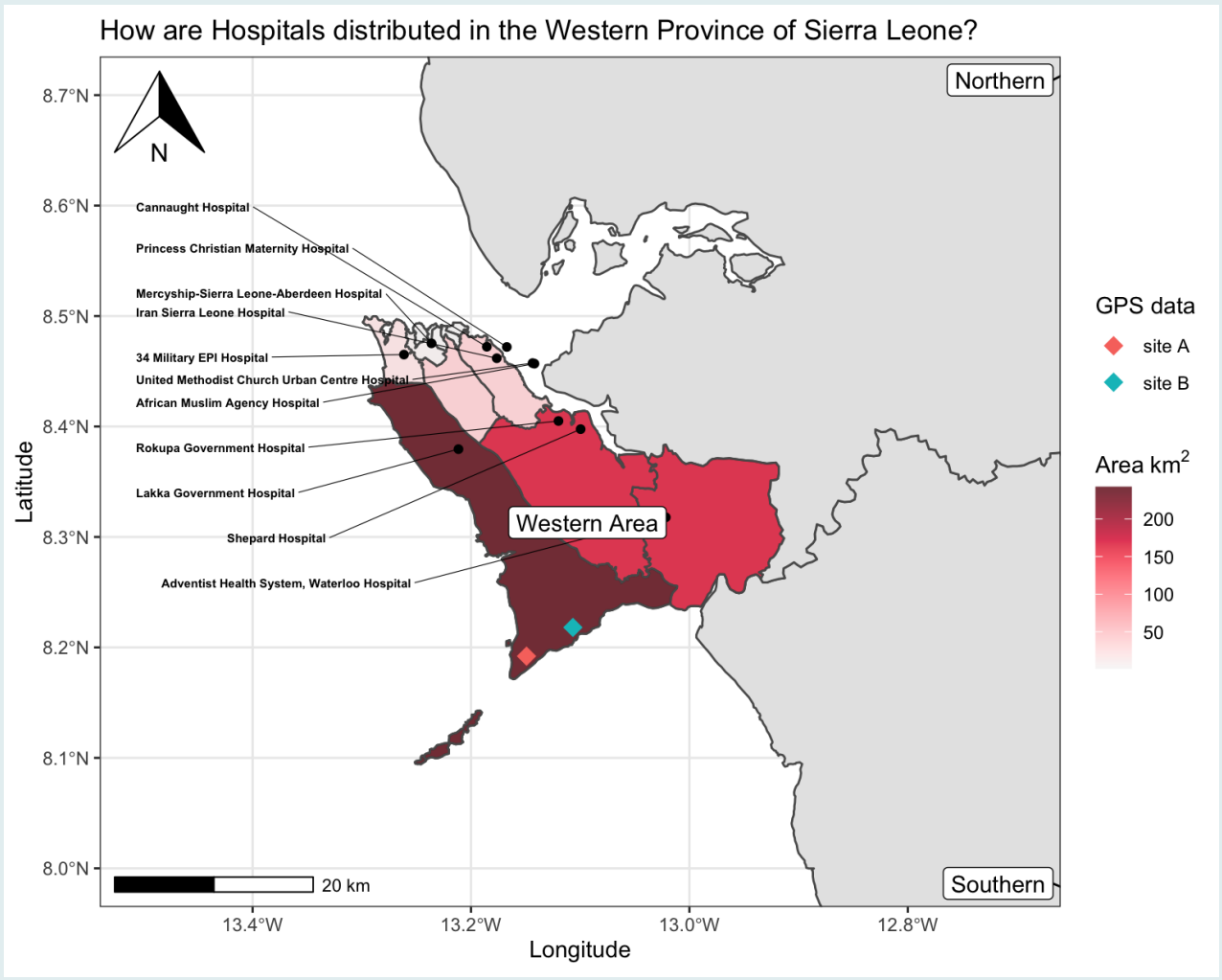
```
  geom_sf() +         # label
  geom_sf_text_repel(mapping = aes(label= facility_name),
                     size = 3,                    fontface = "bold",
                     force = 40,                     box.padding = 0.6)
```



22.5 Final map

For the final map, we put everything together, having a general background map based on the Sierra Leona map, with district delineations, province labels, main hospital names and locations, custom GPS collected field sites, as well as a theme adjusted with axis labels, and a north arrow and scale bar:

```
ggplot() +
  ## geometries
  # background map
  geom_sf(data = sierra_leone) +
  # district polygons filled by area
  geom_sf(data = sierra_leone_shp,
          mapping = aes(fill = area_km2)) +
  # hospital points
  geom_sf(data = sle_healthsites_set) +
  # field site points
  geom_sf(data = sites_sf,
          mapping = aes(color = gps_name),
          shape = 18,
          size = 4) +
  ## labels
  # hospital names with repelled text
  geom_sf_text_repel(data = sle_healthsites_set,
                     mapping = aes(label = facility_name),
                     size         = 2,
                     fontface     = "bold",
                     box.padding  = 0.6,
                     force        = 0.5,
                     nudge_x      = -0.25,
                     direction    = "y",
                     hjust        = 1,
                     segment.size = 0.2) +
  # province names with repelled labels
  geom_sf_label_repel(data = sierra_leone %>%
                        filter(shapeName!="Eastern"),
                      mapping = aes(label=shapeName)) +
  ## aesthetics
  # alternative color scale for fill
  colorspace::scale_fill_continuous_sequential(palette="Reds 3",
                                               alpha = 0.8) +
  # map annotation
  annotation_north_arrow(location="tl") +
  annotation_scale(location="bl") +
  # map extent
  coord_sf(xlim = c(-13.5,-12.7),
           ylim = c(8.0,8.7)) +
  # ggplot labels
  labs(x = "Longitude",
       y = "Latitude",
       fill = expression(Area~km^2),
       color = "GPS data",
       title = "How are Hospitals distributed in the Western Province of
        Sierra Leone?")
```



22.6 Wrap up

This example fully demonstrates that adding **layers** on `ggplot2` is relatively straightforward, as long as the data is properly stored in an `sf` object. Adding additional layers like **external** data, color **palettes**, point or polygon **labels** and **map annotations** would simply follow the same logic, with additional calls *after* `geom_sf()` and at the *right place* in the `ggplot2` sequence.

Contributors

The following team members contributed to this lesson:

ANDREE VALLE CAMPOS

R Developer and Instructor, The Graph Network
Motivated by reproducible science and education.

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education.

References

Some material in this lesson was adapted from the following sources:

- *Moreno, M., Basille, M. Drawing beautiful maps programmatically with R, sf and ggplot2 – Part 1: Basics*. (2018). Retrieved 01 June 2022, from https://r-spatial.org/r/2018/10/25/ggplot2-sf.html
- *Moreno, M., Basille, M. Drawing beautiful maps programmatically with R, sf and ggplot2 – Part 2: Layers*. (2018). Retrieved 01 June 2022, from https://r-spatial.org/r/2018/10/25/ggplot2-sf-2.html
- *Wilke, Claus O. Fundamentals of Data Visualization. Chapter 4: Color scales*. (2020). Retrieved 01 June 2022, from https://clauswilke.com/dataviz/color-basics.html#color-to-represent-data-values