COMP2501 Assignment 1

## Requirements

**Submission deadline: October 9th, 2025 at 23:59 (HKT).**

**Full mark of assignment 1: 50.**

For the following questions, please:

1. Replace all [Input here] places with your information or your answer (for multiple choice).
2. Complete the code block by adding your own code to fulfill the requirements in each question. Please use the existing code block and do not add your own code block.

Please make sure your Rmd file is a valid Markdown document and can be successfully knitted.

For assignment submission, please knit your final Rmd file into a Word document, and submit both your **Rmd** file and the knitted **Microsoft Word** document file to Moodle. You get 0 score if 1) the Rmd file you submitted cannot be knitted, and 2) you have not submitted a Word document. For each visualization question, please make sure that the generated plot is shown in-place with the question and after the code block.

## Name and UID

Name: Shen Hongshan

UID: 3036290936

### Environmental setup

You need to have the dslabs, dplyr, ggplot2, and lubridate packages installed. If not yet, please run install.packages(c("dslabs", "dplyr", "ggplot2", "lubridate")) in your R environment. If you have installed the tidyverse package, dplyr is installed by default.

# Load the required packages  
# install.packages(c("dslabs", "dplyr", "ggplot2", "lubridate"))  
library(dslabs)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

### Example question. Print the first 2 records of the murders dataset.

data("murders")  
head(murders, 2)

## state abb region population total  
## 1 Alabama AL South 4779736 135  
## 2 Alaska AK West 710231 19

### 1. (1 points) Define a matrix mat with 10 rows and 5 columns, print the entries of rows 5-8 and columns 2 & 5 simultaneously.

mat <- matrix(1:50, 10, 5)  
mat[5:8, c(2, 5)]

## [,1] [,2]  
## [1,] 15 45  
## [2,] 16 46  
## [3,] 17 47  
## [4,] 18 48

### 2. (1 points) Write a function compute\_s\_n that for any given n, computes the . Print for from 10 to 20. (Hint: use sapply)

compute\_s\_n <- function(n){  
 n\*log(x = n+5)/(n^(1/2))  
}  
sapply(10:20, compute\_s\_n)

## [1] 8.563607 9.195636 9.814539 10.421384 11.017082 11.602421 12.178090  
## [8] 12.744695 13.302775 13.852815 14.395250

### 3. (2 points) Examine the built-in datasets ChickWeight and co2. Which of the following is true?

* 1. tidy data: all small datasets are tidy by definition.
  2. tidy data: it has one observation for each row.
  3. not tidy: there are multiple rows corresponding to the same observation.
  4. not tidy: it is a matrix instead of a data frame.

ChickWeight is b

co2 is d

### 4. (12 points) Utilize the murders dataset and execute the following tasks.

library(dslabs)  
library(dplyr)  
library(ggplot2)  
data("murders")

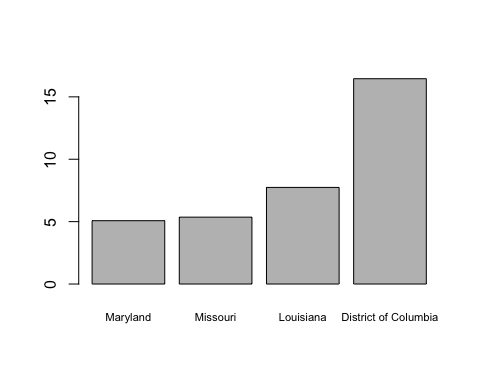
#### a. (2 points) Compute the murder rate per 100,000 people for each state and store it in a column called “rate”. Then use logical operators to find which state has a murder rate per 100,000 people higher than 5. Find these states, print their state names, abbreviations, total populations, and murder rates.

murders |>   
 mutate(rate = total/population\*10^5) |>   
 filter(rate>5) |>   
 select(state, abb, total, rate)

## state abb total rate  
## 1 District of Columbia DC 99 16.452753  
## 2 Louisiana LA 351 7.742581  
## 3 Maryland MD 293 5.074866  
## 4 Missouri MO 321 5.359892

#### b. (2 points) For all states having a murder rate per 100,000 people higher than 5, use the barplot function to create a barplot with the x-axis being the state name, and the y-axis being the murder rate per 100,000 people of each state. Order them by their murder rates. (Hints: check some barplot examples at <https://r-graph-gallery.com/210-custom-barplot-layout.html>)

# create dummy data  
data <- data.frame(  
 murders |> mutate(rate = total/population\*10^5) |> filter(rate>5) |> arrange(rate)  
)  
barplot(height = data$rate, names=data$state, cex.names = 0.7)



#### c. (3 points) Create a table called my\_states that contains rows for states satisfying two conditions: 1) population is lower than 5 million, and 2) the murder rate per 100,000 people is less than 3.0. Use select to show only their state abbreviations, populations, and murder rates. Then use top\_n function to find the 3 safest states among them.

murders |>   
 mutate(rate = total/population\*10^5) |>   
 filter(population<5000000 & rate < 3.0) |>   
 select(abb, total, rate) |>   
 top\_n(-3)

## Selecting by rate

## abb total rate  
## 1 HI 7 0.5145920  
## 2 NH 5 0.3798036  
## 3 VT 2 0.3196211

#### d. (2 points) Categorize all states into two groups based on their population: Above median population – states with a population greater than the median. Below median population – states with a population less than or equal to the median. For each group, calculate the average murder rate per 100,000 people and display the results.

murders |>   
 mutate(population\_group = ifelse(population <= median(population), "Below median population", "Above median population")) |>   
 group\_by(population\_group) |>   
 summarize(average\_murder\_rate = sum(total)/sum(population)\*10^5)

## # A tibble: 2 × 2  
## population\_group average\_murder\_rate  
## <chr> <dbl>  
## 1 Above median population 3.20  
## 2 Below median population 2.21

#### e. (3 points) Use the ggplot2 package to create a scatterplot from the murders dataset, where the x-axis is the number of population, the y-axis is the murder rate per 100,000 people, and each point in the scatterplot is labeled with the state name. Please add an appropriate title, and axis labels to the plot. Select a theme that you prefer.

options(repos = c(CRAN = "https://cloud.r-project.org"))  
install.packages("ggrepel")

## Installing package into '/Users/sunnie/Library/R/arm64/4.5/library'  
## (as 'lib' is unspecified)

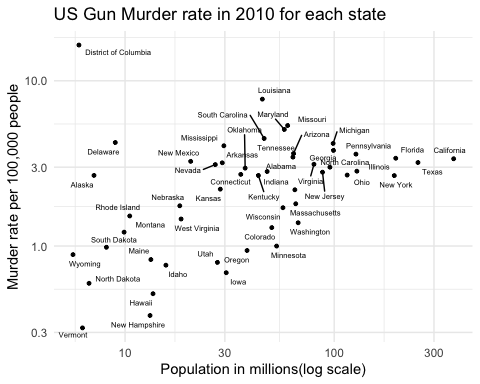
##   
## The downloaded binary packages are in  
## /var/folders/xj/k7cpd6617tx6x3\_06m7dtk280000gn/T//Rtmp5H8lTS/downloaded\_packages

install.packages("ggthemes")

## Installing package into '/Users/sunnie/Library/R/arm64/4.5/library'  
## (as 'lib' is unspecified)

##   
## The downloaded binary packages are in  
## /var/folders/xj/k7cpd6617tx6x3\_06m7dtk280000gn/T//Rtmp5H8lTS/downloaded\_packages

library("ggrepel")  
library("ggthemes")  
  
murders |>   
 ggplot(aes(population/10^5, total/population\*10^5)) +   
 geom\_point(size = 1) +   
 geom\_text\_repel(aes(label = state), max.overlaps = Inf, size = 2) +   
 scale\_x\_log10() +  
 scale\_y\_log10() +  
 labs(x="Population in millions(log scale)", y="Murder rate per 100,000 people", title="US Gun Murder rate in 2010 for each state") +  
 theme\_minimal()



### 5. (4 points) Factors affecting life expectancy with gapminder

library(dslabs)  
library(dplyr)  
library(ggplot2)  
data("gapminder")

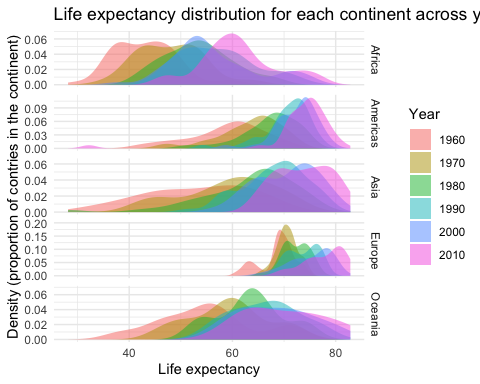
#### a. (2 points) Calculate the average life expectancy and GDP per capita for each continent in 2000. (Hint: Select data from year 2000, group by **continent** and compute for each continent: (i) average **life\_expectancy**, and (ii) **gdp** per capita.) Sort the results by average life expectancy.

gapminder |>  
 filter(year==2000) |>  
 group\_by(continent) |>  
 summarize(average\_life\_expectancy = sum(life\_expectancy\*population)/sum(population),  
 GDP\_per\_capita = sum(gdp)/sum(population)) |>  
 arrange(average\_life\_expectancy)

## # A tibble: 5 × 3  
## continent average\_life\_expectancy GDP\_per\_capita  
## <fct> <dbl> <dbl>  
## 1 Africa 57.2 738.  
## 2 Asia 67.4 2462.  
## 3 Europe 73.9 12742.  
## 4 Americas 74.3 15206.  
## 5 Oceania 74.7 15697.

#### b. (2 points) Show the **distribution** of life expectancy for each continent. Use an appropriate plot (**density** or **histogram**).

gapminder |>  
 filter(year %in% c(1960, 1970, 1980, 1990, 2000, 2010)) |>  
 mutate(year\_factor = as.factor(year)) |>  
 ggplot(aes(life\_expectancy)) +  
 geom\_density(  
 adjust = 1,   
 alpha = 0.5,   
 aes(fill = year\_factor),  
 color = NA) +  
 facet\_grid(continent~., scales="free\_y") +  
 labs(  
 title = "Life expectancy distribution for each continent across years",  
 x = "Life expectancy",  
 y = "Density (proportion of contries in the continent)",  
 fill = "Year"  
 ) +  
 theme\_minimal()



### 6. (7 points) Tropical cyclones with storms (from **dplyr**)

library(dplyr)  
library(ggplot2)  
library(lubridate) # for timestamps  
data("storms")

#### a. (1 point) List the observation with the **largest wind** speed.

storms |> filter(wind == max(wind))

## # A tibble: 1 × 13  
## name year month day hour lat long status category wind pressure  
## <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct> <dbl> <int> <int>  
## 1 Allen 1980 8 7 18 21.8 -86.4 hurricane 5 165 899  
## # ℹ 2 more variables: tropicalstorm\_force\_diameter <int>,  
## # hurricane\_force\_diameter <int>

#### b. (2 points) For (1) the whole dataset, (2) by **year**, compute both: (i) the **maximum wind** and (ii) the **minimum pressure** observed. For the per-year stats, sort the results by maximum wind (descending) and show the top 10 rows.

# (1) Whole dataset  
storms |>  
 summarise(max\_wind = max(wind), min\_pressure = min(pressure))

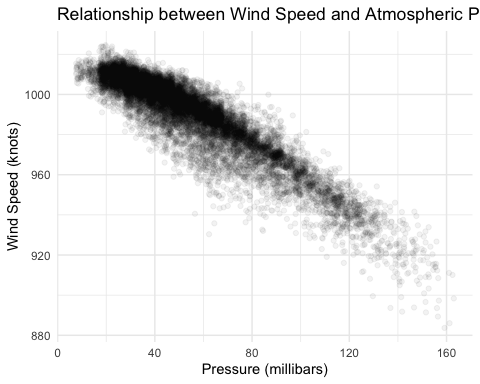
## # A tibble: 1 × 2  
## max\_wind min\_pressure  
## <int> <int>  
## 1 165 882

# (2) By year  
storms |>  
 group\_by(year) |>  
 summarise(max\_wind = max(wind), min\_pressure = min(pressure)) |>  
 arrange(desc(max\_wind)) |>  
 top\_n(10, max\_wind)

## # A tibble: 10 × 3  
## year max\_wind min\_pressure  
## <dbl> <int> <int>  
## 1 1980 165 899  
## 2 1988 160 888  
## 3 2005 160 882  
## 4 2019 160 910  
## 5 1998 155 905  
## 6 2017 155 908  
## 7 1977 150 926  
## 8 1979 150 924  
## 9 1992 150 922  
## 10 2007 150 905

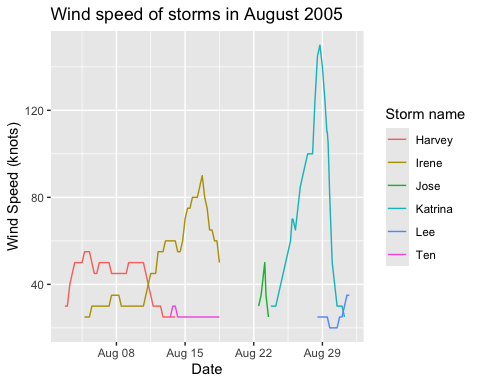
#### c. (1 point) Create a **scatter plot** of **wind** vs **pressure**. Note that the wind speeds are recorded in knots and pressures are recorded in millibars. Add a title and axis labels.

storms |>  
 ggplot(aes(wind, pressure)) +  
 geom\_jitter(alpha = 0.05, width = 2.5, height = 2) +  
 labs(  
 title = "Relationship between Wind Speed and Atmospheric Pressure in Storms",  
 x = "Pressure (millibars)",  
 y = "Wind Speed (knots)"  
 ) +  
 theme\_minimal()



#### d. (3 points) Create a **multi-line plot** for all storms from August, 2005, showing **timestamp** on x-axis and **wind** on y-axis, with color indicating **storm name**. Add a legend, title, and labels. (Hint: use functions from the lubridate package to define timestamps)

data <- storms |>   
 filter(year == 2005 & month == 8) |>  
 mutate(timestamp = make\_datetime(year, month, day, hour)) |>  
 arrange(name, timestamp)  
data |>   
 ggplot(aes(timestamp, wind, color = name)) +  
 geom\_line() +  
 labs(  
 title = "Wind speed of storms in August 2005",  
 x = "Date",  
 y = "Wind Speed (knots)",  
 color = "Storm name"  
 )



### 7. (23 points) Explore the tidyverse with the COVID-19 dataset (<http://www.bio8.cs.hku.hk/comp2501/covid.csv>), and answer the following questions. Noticeably, it is acceptable to analyze the data or generate a plot with negative values of cases and deaths.

library(dplyr)  
library(ggplot2)  
library(lubridate)

#### a. (2 points) Read the CSV formatted dataset. Find out how many observations (rows) and variables (columns) are in the dataset. Print the names of all variables.

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ tibble 3.3.0 ✔ purrr 1.1.0  
## ✔ tidyr 1.3.1 ✔ stringr 1.5.2  
## ✔ readr 2.1.5 ✔ forcats 1.0.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ lubridate::as.difftime() masks base::as.difftime()  
## ✖ lubridate::date() masks base::date()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ lubridate::intersect() masks base::intersect()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ lubridate::setdiff() masks base::setdiff()  
## ✖ lubridate::union() masks base::union()

covid <- read\_csv("http://www.bio8.cs.hku.hk/comp2501/covid.csv")

## Rows: 47480 Columns: 12  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (5): dateRep, countriesAndTerritories, geoId, countryterritoryCode, cont...  
## dbl (7): day, month, year, cases, deaths, popData2019, Cumulative\_number\_for...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

nrow(covid)

## [1] 47480

names(covid)

## [1] "dateRep"   
## [2] "day"   
## [3] "month"   
## [4] "year"   
## [5] "cases"   
## [6] "deaths"   
## [7] "countriesAndTerritories"   
## [8] "geoId"   
## [9] "countryterritoryCode"   
## [10] "popData2019"   
## [11] "continentExp"   
## [12] "Cumulative\_number\_for\_14\_days\_of\_COVID-19\_cases\_per\_100000"

#### b. (1 points) List 5 observations that have the most amounts of deaths per day.

covid |> arrange(desc(deaths)) |> top\_n(5, deaths)

## # A tibble: 5 × 12  
## dateRep day month year cases deaths countriesAndTerritories geoId  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr>  
## 1 16/04/2020 16 4 2020 30148 4928 United\_States\_of\_America US   
## 2 14/08/2020 14 8 2020 9441 3935 Peru PE   
## 3 24/07/2020 24 7 2020 4546 3887 Peru PE   
## 4 07/09/2020 7 9 2020 -8261 3800 Ecuador EC   
## 5 18/04/2020 18 4 2020 30833 3770 United\_States\_of\_America US   
## # ℹ 4 more variables: countryterritoryCode <chr>, popData2019 <dbl>,  
## # continentExp <chr>,  
## # `Cumulative\_number\_for\_14\_days\_of\_COVID-19\_cases\_per\_100000` <dbl>

#### c. (2 points) How many unique countriesAndTerritories are in the dataset?

unique(covid$"countriesAndTerritories") |> length()

## [1] 210

#### d. (3 points) For (1) the whole dataset, (2) different countriesAndTerritories, and (3) different continentExp, compute both (i) the sum of cases, and (ii) the sum of deaths. Sort the results by the sum of deaths descendingly. Use head to show the top 3 entries if there are too many rows.

# (1) Whole dataset  
covid |>  
 summarise(sum\_of\_cases = sum(cases), sum\_of\_deaths = sum(deaths))

## # A tibble: 1 × 2  
## sum\_of\_cases sum\_of\_deaths  
## <dbl> <dbl>  
## 1 35848254 1048181

# (2) By countries and territories  
covid |>  
 group\_by(countriesAndTerritories) |>  
 summarise(sum\_of\_cases = sum(cases), sum\_of\_deaths = sum(deaths)) |>  
 arrange(desc(sum\_of\_deaths)) |>  
 head(3)

## # A tibble: 3 × 3  
## countriesAndTerritories sum\_of\_cases sum\_of\_deaths  
## <chr> <dbl> <dbl>  
## 1 United\_States\_of\_America 7501612 210909  
## 2 Brazil 4969141 147494  
## 3 India 6757131 104555

# (3) By continents  
covid |>  
 group\_by(continentExp) |>  
 summarise(sum\_of\_cases = sum(cases), sum\_of\_deaths = sum(deaths)) |>  
 arrange(desc(sum\_of\_deaths))

## # A tibble: 6 × 3  
## continentExp sum\_of\_cases sum\_of\_deaths  
## <chr> <dbl> <dbl>  
## 1 America 17445678 578079  
## 2 Europe 5605508 228689  
## 3 Asia 11233759 203583  
## 4 Africa 1528213 36828  
## 5 Oceania 34400 995  
## 6 Other 696 7

# only 6 rows, so `|> head(3)` is unnecessary

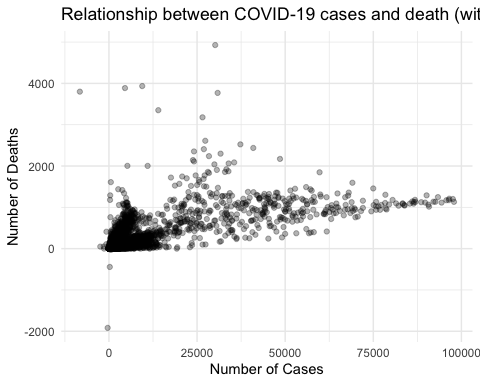
#### e. (1 points) Add a new column date with the standard date format “YYYY-MM-DD” to the data table according to the dateRep column. Be reminded the format of dateRep is “DD/MM/YYYY”. Please use head() to show the result. (Hint: use functions from the lubridate package to read and write timestamps)

covid <- mutate(covid, date = dmy(dateRep))  
head(covid)

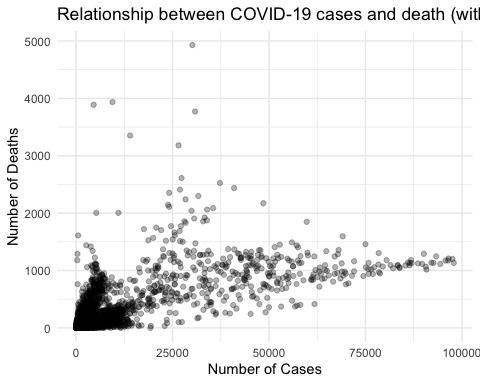
## # A tibble: 6 × 13  
## dateRep day month year cases deaths countriesAndTerritories geoId  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr>  
## 1 07/10/2020 7 10 2020 62 2 Afghanistan AF   
## 2 06/10/2020 6 10 2020 145 5 Afghanistan AF   
## 3 05/10/2020 5 10 2020 44 0 Afghanistan AF   
## 4 04/10/2020 4 10 2020 7 4 Afghanistan AF   
## 5 03/10/2020 3 10 2020 5 0 Afghanistan AF   
## 6 02/10/2020 2 10 2020 17 0 Afghanistan AF   
## # ℹ 5 more variables: countryterritoryCode <chr>, popData2019 <dbl>,  
## # continentExp <chr>,  
## # `Cumulative\_number\_for\_14\_days\_of\_COVID-19\_cases\_per\_100000` <dbl>,  
## # date <date>

#### f. (2 points) Create a scatterplot showing cases vs. deaths. Set an appropriate plot title and axis titles. Are there any dots that seem abnormal? Exclude those and plot again.

# (1) with outliers  
covid |>  
 ggplot(aes(cases, deaths)) +  
 geom\_point(alpha = 0.3) +  
 labs(  
 title = "Relationship between COVID-19 cases and death (with outliers)",  
 x = "Number of Cases",  
 y = "Number of Deaths"  
 ) +  
 theme\_minimal()

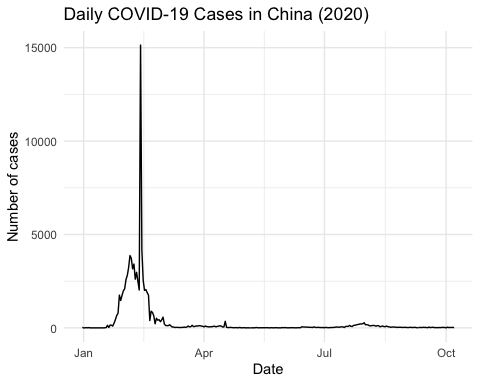


# (2) without outliers  
covid\_clean <- filter(covid, cases >= 0 & deaths >= 0)  
covid\_clean |>  
 filter(cases >=0 & deaths >= 0) |>  
 ggplot(aes(cases, deaths)) +  
 geom\_point(alpha = 0.3) +  
 labs(  
 title = "Relationship between COVID-19 cases and death (without outliers)",  
 x = "Number of Cases",  
 y = "Number of Deaths"  
 ) +  
 theme\_minimal()



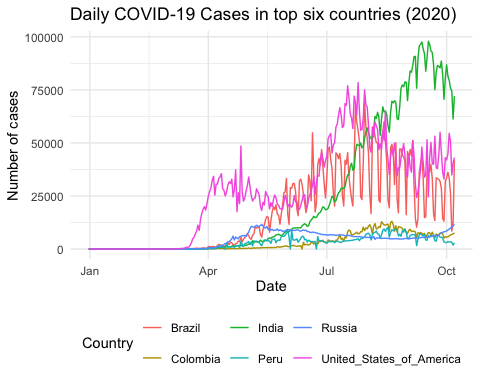
#### g. (2 points) Create a line plot showing cases per day versus date for China. Set an appropriate plot title and axis titles.

covid\_clean |>  
 filter(countriesAndTerritories=="China") |>  
 ggplot(aes(date, cases)) +  
 geom\_line() +  
 labs(  
 title = "Daily COVID-19 Cases in China (2020)",  
 x = "Date",  
 y = "Number of cases"  
 ) +  
 theme\_minimal()



#### h. (2 points) Similar to above, create a line plot using the data of six countries with the most total cases. Use different line colors for each country. Set an appropriate plot title and axis titles.

max\_total\_country <- covid\_clean |>  
 group\_by(countriesAndTerritories) |>  
 summarise(total\_cases = sum(cases)) |>  
 arrange(desc(total\_cases)) |>  
 head() |>  
 \_$countriesAndTerritories  
  
covid\_clean |>  
 filter(countriesAndTerritories %in% max\_total\_country) |>  
 ggplot(aes(as.Date(date), cases, color = countriesAndTerritories, group = countriesAndTerritories)) +  
 geom\_line() +  
 labs(  
 title = "Daily COVID-19 Cases in top six countries (2020)",  
 x = "Date",  
 y = "Number of cases",  
 color = "Country"  
 ) +  
 theme\_minimal() +  
 theme(  
 legend.position = "bottom",  
 legend.direction = "horizontal"  
 )



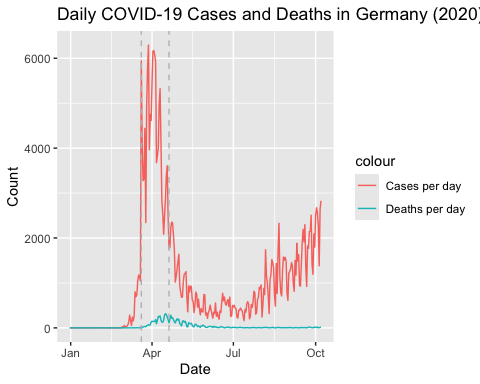
#### i. (4 points) Filter out the data in Germany. Assume the Germany government determined an epidemic situation of national significance on 2020-06-10. A lot of restrictive measurements such as social distancing were implemented to prevent the spread of COVID-19. Find out the 1) average number of cases before the determination, 2) average number of cases after the determination, 3) average number of deaths before the determination, and 4) average number of deaths after the determination. Remove missing values from data if they are coded as NA, if there is any.

covid\_clean |>  
 filter(countriesAndTerritories=="Germany" & is.na(cases)==FALSE & is.na(deaths)==FALSE) |>  
 mutate(period = if\_else(date < "2020-06-10",   
 "Before\_determination", "After\_determination")) |>  
 group\_by(period) |>  
 summarise(average\_cases = mean(cases), average\_deaths = mean(deaths)) |>  
 arrange(desc(period))

## # A tibble: 2 × 3  
## period average\_cases average\_deaths  
## <chr> <dbl> <dbl>  
## 1 Before\_determination 1139. 53.8   
## 2 After\_determination 1013. 7.09

#### j. (4 points) In fact, the determination of epidemic situation was on another date. Make a reasonable guess with explanation by the following steps. Create a line plot using data with countriesAndTerritories=="Germany", showing date on the x-axis, cases per day and death per day on the y-axis. Observe the pattern and make a reasonable guess. Find a way to indicate the date of the determination and 1 month after the date of determination. Set an appropriate plot title and axis titles. Marks would be given on answers with reasonable explanations. (Hints: It is expected that the cases confirmed would drop after 1 month of the implementation of determination of epidemic situation, with 14-week incubation period of COVID-19. It is a relatively open question, and reasonable explanations are acceptable. We want you to observe the trend of changes in the cases per day and the deaths per day on the date you have chosen and one month after it according to the line plot you created. No correct date is needed and we will judge your answers according to your thoughts and explanations.)

covid\_clean |>  
 filter(countriesAndTerritories=="Germany" & is.na(cases)==FALSE & is.na(deaths)==FALSE) |>  
 ggplot(aes(x = as.Date(date))) +  
 geom\_line(aes(y = cases, color = "Cases per day")) +  
 geom\_line(aes(y = deaths, color = "Deaths per day")) +  
 labs(  
 title = "Daily COVID-19 Cases and Deaths in Germany (2020)",  
 x = "Date",  
 y = "Count",  
 ) +  
 geom\_vline(  
 xintercept = c(as.Date("2020-03-20"), as.Date("2020-04-20")),  
 linetype = "dashed",  
 color = "gray",  
 )



theme\_minimal() +  
 theme(legend.position = "bottom", legend.direction = "horizontal")

## <theme> List of 144  
## $ line : <ggplot2::element\_line>  
## ..@ colour : chr "black"  
## ..@ linewidth : num 0.5  
## ..@ linetype : num 1  
## ..@ lineend : chr "butt"  
## ..@ linejoin : chr "round"  
## ..@ arrow : logi FALSE  
## ..@ arrow.fill : chr "black"  
## ..@ inherit.blank: logi TRUE  
## $ rect : <ggplot2::element\_rect>  
## ..@ fill : chr "white"  
## ..@ colour : chr "black"  
## ..@ linewidth : num 0.5  
## ..@ linetype : num 1  
## ..@ linejoin : chr "round"  
## ..@ inherit.blank: logi TRUE  
## $ text : <ggplot2::element\_text>  
## ..@ family : chr ""  
## ..@ face : chr "plain"  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : chr "black"  
## ..@ size : num 11  
## ..@ hjust : num 0.5  
## ..@ vjust : num 0.5  
## ..@ angle : num 0  
## ..@ lineheight : num 0.9  
## ..@ margin : <ggplot2::margin> num [1:4] 0 0 0 0  
## ..@ debug : logi FALSE  
## ..@ inherit.blank: logi TRUE  
## $ title : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : NULL  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ point : <ggplot2::element\_point>  
## ..@ colour : chr "black"  
## ..@ shape : num 19  
## ..@ size : num 1.5  
## ..@ fill : chr "white"  
## ..@ stroke : num 0.5  
## ..@ inherit.blank: logi TRUE  
## $ polygon : <ggplot2::element\_polygon>  
## ..@ fill : chr "white"  
## ..@ colour : chr "black"  
## ..@ linewidth : num 0.5  
## ..@ linetype : num 1  
## ..@ linejoin : chr "round"  
## ..@ inherit.blank: logi TRUE  
## $ geom : <ggplot2::element\_geom>  
## ..@ ink : chr "black"  
## ..@ paper : chr "white"  
## ..@ accent : chr "#3366FF"  
## ..@ linewidth : num 0.5  
## ..@ borderwidth: num 0.5  
## ..@ linetype : int 1  
## ..@ bordertype : int 1  
## ..@ family : chr ""  
## ..@ fontsize : num 3.87  
## ..@ pointsize : num 1.5  
## ..@ pointshape : num 19  
## ..@ colour : NULL  
## ..@ fill : NULL  
## $ spacing : 'simpleUnit' num 5.5points  
## ..- attr(\*, "unit")= int 8  
## $ margins : <ggplot2::margin> num [1:4] 5.5 5.5 5.5 5.5  
## $ aspect.ratio : NULL  
## $ axis.title : NULL  
## $ axis.title.x : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : num 1  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 2.75 0 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.title.x.top : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : num 0  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 0 2.75 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.title.x.bottom : NULL  
## $ axis.title.y : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : num 1  
## ..@ angle : num 90  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 2.75 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.title.y.left : NULL  
## $ axis.title.y.right : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : num 1  
## ..@ angle : num -90  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 0 0 2.75  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : chr "#4D4D4DFF"  
## ..@ size : 'rel' num 0.8  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : NULL  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.x : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : num 1  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 2.2 0 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.x.top : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 0 4.95 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.x.bottom : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 4.95 0 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.y : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : num 1  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 2.2 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.y.left : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 4.95 0 0  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.y.right : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 0 0 4.95  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.text.theta : NULL  
## $ axis.text.r : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : num 0.5  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : <ggplot2::margin> num [1:4] 0 2.2 0 2.2  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ axis.ticks : <ggplot2::element\_blank>  
## $ axis.ticks.x : NULL  
## $ axis.ticks.x.top : NULL  
## $ axis.ticks.x.bottom : NULL  
## $ axis.ticks.y : NULL  
## $ axis.ticks.y.left : NULL  
## $ axis.ticks.y.right : NULL  
## $ axis.ticks.theta : NULL  
## $ axis.ticks.r : NULL  
## $ axis.minor.ticks.x.top : NULL  
## $ axis.minor.ticks.x.bottom : NULL  
## $ axis.minor.ticks.y.left : NULL  
## $ axis.minor.ticks.y.right : NULL  
## $ axis.minor.ticks.theta : NULL  
## $ axis.minor.ticks.r : NULL  
## $ axis.ticks.length : 'rel' num 0.5  
## $ axis.ticks.length.x : NULL  
## $ axis.ticks.length.x.top : NULL  
## $ axis.ticks.length.x.bottom : NULL  
## $ axis.ticks.length.y : NULL  
## $ axis.ticks.length.y.left : NULL  
## $ axis.ticks.length.y.right : NULL  
## $ axis.ticks.length.theta : NULL  
## $ axis.ticks.length.r : NULL  
## $ axis.minor.ticks.length : 'rel' num 0.75  
## $ axis.minor.ticks.length.x : NULL  
## $ axis.minor.ticks.length.x.top : NULL  
## $ axis.minor.ticks.length.x.bottom: NULL  
## $ axis.minor.ticks.length.y : NULL  
## $ axis.minor.ticks.length.y.left : NULL  
## $ axis.minor.ticks.length.y.right : NULL  
## $ axis.minor.ticks.length.theta : NULL  
## $ axis.minor.ticks.length.r : NULL  
## $ axis.line : <ggplot2::element\_blank>  
## $ axis.line.x : NULL  
## $ axis.line.x.top : NULL  
## $ axis.line.x.bottom : NULL  
## $ axis.line.y : NULL  
## $ axis.line.y.left : NULL  
## $ axis.line.y.right : NULL  
## $ axis.line.theta : NULL  
## $ axis.line.r : NULL  
## $ legend.background : <ggplot2::element\_blank>  
## $ legend.margin : NULL  
## $ legend.spacing : 'rel' num 2  
## $ legend.spacing.x : NULL  
## $ legend.spacing.y : NULL  
## $ legend.key : <ggplot2::element\_blank>  
## $ legend.key.size : 'simpleUnit' num 1.2lines  
## ..- attr(\*, "unit")= int 3  
## $ legend.key.height : NULL  
## $ legend.key.width : NULL  
## $ legend.key.spacing : NULL  
## $ legend.key.spacing.x : NULL  
## $ legend.key.spacing.y : NULL  
## $ legend.key.justification : NULL  
## $ legend.frame : NULL  
## $ legend.ticks : NULL  
## $ legend.ticks.length : 'rel' num 0.2  
## $ legend.axis.line : NULL  
## $ legend.text : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : 'rel' num 0.8  
## ..@ hjust : NULL  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : NULL  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ legend.text.position : NULL  
## $ legend.title : <ggplot2::element\_text>  
## ..@ family : NULL  
## ..@ face : NULL  
## ..@ italic : chr NA  
## ..@ fontweight : num NA  
## ..@ fontwidth : num NA  
## ..@ colour : NULL  
## ..@ size : NULL  
## ..@ hjust : num 0  
## ..@ vjust : NULL  
## ..@ angle : NULL  
## ..@ lineheight : NULL  
## ..@ margin : NULL  
## ..@ debug : NULL  
## ..@ inherit.blank: logi TRUE  
## $ legend.title.position : NULL  
## $ legend.position : chr "bottom"  
## $ legend.position.inside : NULL  
## $ legend.direction : chr "horizontal"  
## $ legend.byrow : NULL  
## $ legend.justification : chr "center"  
## $ legend.justification.top : NULL  
## $ legend.justification.bottom : NULL  
## $ legend.justification.left : NULL  
## $ legend.justification.right : NULL  
## $ legend.justification.inside : NULL  
## [list output truncated]  
## @ complete: logi TRUE  
## @ validate: logi TRUE

Guess the date of determination: around 2020-03-20

Explanation:

* before determination(~ 2020-03-20): quick growth trend in daily confirmed cases, which aligns with a lenient containment measures, no control.
* determination to 1 month later (2020-03-20 ~ 2020-04-20): change of daily cases over time, shift from rapid growth to stabilization; Peak is reached in early April followed by gradual decline; Deaths’ rate of growth slowed down, indicating the determination
* after 1 month (2020-04-20 ~): obvious declining trend in both number of daily cases and deaths

Above trends aligns with the fact that: the cases confirmed would drop after 1 month of the implementation of determination of epidemic situation, with 14-week incubation period of COVID-19.