Using Digital Footprint Data to Measure and Monitor Human Mobility

Prof. Francisco Rowe, Dr. Carmen Cabrera-Arnau, Dr. Miguel Gonzalez-Leonardo, R

7/7/23

Table of contents

Welcome

This website hosts the materials for the workshop *"Using Digital Footprint Data to Measure and Monitor Human Mobility"*. This training workshop was designed and is delivered by Prof. Francisco Rowe, Dr. Carmen Cabrera-Arnau, Dr. Miguel González-Leonardo, Ruth Neville and Andrea Nasuto.

Contact

Prof. Francisco Rowe, Professor in Population Data Science
f.rowe-gonzalez [at] liverpool.ac.uk
Department of Geography and Planning, University of Liverpool, Liverpool, UK

Dr Carmen Cabrera-Arnau, Lecturer in Geographic Data Science
c.cabrera-arnau [at] liverpool.ac.uk
Department of Geography and Planning, University of Liverpool, Liverpool, UK

Dr Miguel Gonzalez-Leonardo, Assistant Professor
miguel.gonzalez [at] colmex.mx
Center for Demographic, Urban and Environmental Studies, El Colegio de Mexico, Mexico City, Mexico

Ruth Neville, PhD candidate
ruth.neville [at] liverpool.ac.uk
Department of Geography and Planning, University of Liverpool, Liverpool, UK

Andrea Nasuto, PhD candidate
andrea.nasuto [at] liverpool.ac.uk
Department of Geography and Planning, University of Liverpool, Liverpool, UK

Overview

Description

This workshop offers an introduction to the use of digital footprint data from Meta-Facebook to analyse internal population movements. We will cover how to use location data collected via digital technology to develop data products to analyse human mobility. We will also illustrate how these data can be used to create statistical indicators and geographic data visualisations to extract insightful information, monitor mobility trends and identify key areas of origin and destination. The workshop will use an applied, hands-on approach in R via computational notebooks.

Structure

Time	Activity
9.00-9.30	Registration and welcome coffee
9.30-10.30	Digital Footprint Data overview
10.30-10.40	Comfort break
10.40-11.40	Meta-Facebook data introduction
11.40-12.15	Software installation and preparation for practical session
12.15-13.15	Lunch break
13.15-14.30	Exploring temporal patterns of human mobility
14.30-15.00	Coffee break
15.00-16.30	Exploring spatial patterns of human mobility
16.30-17.00	Q&A

Before the workshop

! Important

Please make sure you download and install the most recent version of R, RStudio and Quarto on the computer that you will be using during the workshop, and install the indicated R packages – see detailed instructions below.

ℹ Note

All three software packages are open and free to use.

R

You can download R here. Make sure you select the appropriate version for your Operating System: Windows, MacOS (Apple silicon M1/M2 or older intel Macs). For example, if you use a macOS laptop with an M1 processor, click on 'Download R for macOS' and then, click the link to download the installer file (.pkg extension for macOS) under the header 'For Apple silicon (M1/M2) Macs'. You can then open the installer and follow the instructions that you will be prompted with. For Windows users, click on 'install R for the first time' and follow the prompts.

RStudio

You will also need to download RStudio Desktop (or simply RStudio), which is an integrated development environment to help you write code in R more easily. To download RStudio, follow this link and scroll down to the section titled 'All Installers and Tarballs'. Download the appropriate installer file according to your Operating System. Then, open the installer and follow the installation instructions that you will be prompted with.

Quarto

Download Quarto from this website. Quarto is a publishing system that will allow you to open and work on the computational notebooks for the workshop. On 'Step 1' on the website, download the version of Quarto that matches your Operating System. Open the installer file, run it and follow the prompts.

R libraries

Once you have installed R, you will need to install some R extensions, known as packages, that will be useful for the applications explored in this workshop. The packages you need to install are:

- `tidyverse`
- `ggthemes`

- `patchwork`
- `viridis`
- `tmap`
- `sf`
- `sp`
- `stringr`
- `RColorBrewer`

To install the packages, open RStudio. On the console window (normally at the bottom left), write the following command: install.packages("name of package"). Make sure you replace "name of package" by the actual name of the package that you want to install e.g. install.packages("tidyverse"). Then, press enter and repeat this process until you have installed all the packages in the list.

> **Note**
>
> We might ask you to install more packages on the day that this workshop is taking place.

During the workshop

All the workshop material will be made available on this website which is currently under construction. Further instructions on how to download the material will be given during the workshop.

1 Temporal patterns

In this part of the workshop, we explore the temporal patterns displayed by human mobility data from Facebook Meta. The data was collected for Chile in the time period spanning from end of March of 2020 to the end of March 2022.

```
#Before getting started, please run the code below to avoid getting warnings and messages
```

1.1 Dependencies

To analyse the data, we need R and the following R packages (see installation instructions here*). Once installed, you can use the R packages by loading them as libraries:

```
library(dplyr)
library(tidyverse)
library(zoo)
library(ggplot2)
library(lubridate)
library(tools)
library(mice)
```

1.2 Loading and pre-processing the data

We start by looking at the temporal evolution of internal human mobility in Chile in the first few months after the first outbreak of COVID-19. To do so, we read the movement data corresponding to the months of May, June, July, August and September 2020. The data is stored as .rds files and can be read using the `readRDS()` function:

```
df_5 <- readRDS("./data/fb/movement_grid/2020_05_mov.rds")
df_6 <- readRDS("./data/fb/movement_grid/2020_06_mov.rds")
df_7 <- readRDS("./data/fb/movement_grid/2020_07_mov.rds")
df_8 <- readRDS("./data/fb/movement_grid/2020_08_mov.rds")
df_9 <- readRDS("./data/fb/movement_grid/2020_09_mov.rds")
```

Since we are interested in the evolution of trends in internal human mobility over this five-month period, we bind the data sets together using the function `rbind()`. This function appends datasets that share the same columns. We call the resulting data set `df_mov`.

```
df_mov <- rbind(df_5, df_6, df_7, df_8, df_9)
```

But, what are the columns of `df1`, `df2`, `df3`, `df4`, `df5` and `df_mov`? We can easily visualise a dataset and its contents with the `glimpse()` function. For example, for `df_mov`:

```
glimpse(df_mov)
```

```
Rows: 630,635
Columns: 23
$ geometry                      <chr> "LINESTRING (-71.630859375 -36.5272625723~
$ date_time                     <chr> "2020-05-01 08:00:00", "2020-05-01 08:00:~
$ start_polygon_id              <chr> "60871", "60871", "60871", "60871", "6087~
$ start_polygon_name            <chr> "Ñuble", "Ñuble", "Ñuble", "Ñuble", "Ñubl~
$ end_polygon_id                <chr> "60871", "60871", "60871", "60871", "6087~
$ end_polygon_name              <chr> "Ñuble", "Ñuble", "Ñuble", "Ñuble", "Ñubl~
$ length_km                     <dbl> 0.00000, 0.00000, 0.00000, 0.00000, 0.000~
$ tile_size                     <dbl> 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 1~
$ country                       <chr> "CL", "CL", "CL", "CL", "CL", "CL", "CL",~
$ level                         <chr> "LEVEL3", "LEVEL3", "LEVEL3", "LEVEL3", "~
$ n_crisis                      <dbl> 228, 455, 989, 2293, 111, 23, 22, 3045, 1~
$ n_baseline                    <dbl> 236.2, 367.6, 1162.4, 1456.8, 92.6, 45.6,~
$ n_difference                  <dbl> -8.2, 87.4, -173.4, 836.2, 18.4, -22.6, -~
$ percent_change                <dbl> -3.456998, 23.711340, -14.904590, 57.3604~
$ is_statistically_significant <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ z_score                       <dbl> -0.09754166, 1.02989015, -0.47284604, 4.0~
$ start_lat                     <dbl> -36.64353, -36.64353, -36.64353, -36.6435~
$ start_lon                     <dbl> -71.97498, -71.97498, -71.97498, -71.9749~
$ end_lat                       <dbl> -36.64353, -36.64353, -36.64353, -36.6435~
$ end_lon                       <dbl> -71.97498, -71.97498, -71.97498, -71.9749~
$ start_quadkey                 <chr> "21023300110", "21023300110", "2102330011~
$ end_quadkey                   <chr> "21023300110", "21023300110", "2102330011~
$ date_time2                    <dttm> 2020-05-01 07:00:00, 2020-05-01 07:00:00~
```

In the output, we first see the number of rows and columns in the data set. Then we see the name of each column, followed by the type of data stored in that column (e.g. `<chr>` for character), and a sample of the data values. Alternatively, if we only want to know the names of the columns, we can use the `names()` function:

```r
names(df_mov)
```

```
 [1] "geometry"                        "date_time"
 [3] "start_polygon_id"                "start_polygon_name"
 [5] "end_polygon_id"                  "end_polygon_name"
 [7] "length_km"                       "tile_size"
 [9] "country"                         "level"
[11] "n_crisis"                        "n_baseline"
[13] "n_difference"                    "percent_change"
[15] "is_statistically_significant" "z_score"
[17] "start_lat"                       "start_lon"
[19] "end_lat"                         "end_lon"
[21] "start_quadkey"                   "end_quadkey"
[23] "date_time2"
```

As you might have guessed, each column represents a variable or a characteristic of the data stored in the dataset. Each row or data point in the resulting dataset `df_mov` represents a population flow between an origin and a destination location, taking place during a given time window, in this case, within the five-month period starting in May 2020. The information about the origin and destination of the flow is fully determined by the `geometry` column. Other columns also contain information about the start and end locations of the movements, such as `start_polygon_id`, `start_polygon_name`, `end_polygon_id`, `end_polygon_name`, `start_lat`, `start_lon`, `end_lat`, `end_lon`, `start_quadkey` and `end_quadkey`. The information about the time window in which each population flow takes place is contained in the `date_time` column.

In the next few sections, we will focus only on three variables, namely, `date_time`, `n_crisis` and `n_baseline`. But first, let's talk a bit more about the datetime format.

1.2.1 Choice of variables for the analysis

Of all the columns in the dataset `df_mov`, three of them are particularly relevant for the analysis in this section of the workshop, `n_crisis`, `n_baseline` and `date_time`. Firstly, the variable `n_crisis` gives us the size of a given flow, i.e. the number of people moving from an origin to a destination. For the first flow, this would be:

```r
df_mov$n_crisis[[1]]
```

```
[1] 228
```

But is this number large or small? In order to establish a benchmark, we can look at `n_baseline`. This variable tells us about the number of people that we would expect travelling from origin to destination on the same weekday at the same time of the day in normal conditions, i.e. pre-pandemic.

```
df_mov$n_baseline[[1]]
```

```
[1] 236.2
```

The fact that the value of `n_crisis` is smaller than `n_baseline` for the first flow suggests that during the pandemic people were moving less between the particular origin-destination pair of locations corresponding to the first flow in the dataset.

However, in order to draw conclusions at the population level, looking at a data point in isolation is not enough. We can gain much more meaningful insights if we look at much larger amounts of data as well as the temporal evolution of the observed patterns. To do this, we need to look into the `date_time` variable, that tells us the 8-hour time window in which a flow takes place. For example, we can look up the value of `date_time` for the first flow recorded in `df_mov`:

```
df_mov$date_time[[1]]
```

```
[1] "2020-05-01 08:00:00"
```

Therefore, the first flow in the dataset `df_mov` took place between 00:00 and 08:00 of the 1st of May 2020. However, for the analysis in this workbook, we are interested only in the patterns taking place at the daily scale, so the information regarding the time window within the day is irrelevant.

1.2.2 Extracting the calendar date for each population flow

To explore the variation in the number of flows at the daily level, we need to first extract only the date from each movement. This can be done by running the code below, where we create a new column in `df_mov` named `date`. This new column stores only the information corresponding to date which can be obtained by applying the `as.Date()` function to the data from the `date_time` column. This function allows us to convert character data into objects of class `Date` representing calendar dates and following a specified format (in this case, year-month-day).

```
df_mov$date <- as.Date(df_mov$date_time, format = "%Y-%m-%d")
```

1.2.3 Grouping data by calendar date

Since we are interested in the number of movements taking place on a daily basis, we need to aggregate the information from flows that take place on the same day. Specifically, we need to sum all the values of `n_crisis` corresponding to flows from the same day. This can be done, firstly, by passing the `df_mov` dataset to the `group_by()` function and specifying that we want to group the data according to the information stored in the `df_mov$date` column that we just created. This action can be achieved via the pipeline operator `%>%`, which simply feeds the results of one operation into the next operation. For those familiar with the mathematical language, the pipe operator is equivalent to the function composition operation. But the `group_by()` function alone will not give the desired output, so we follow it by the `summarise()` function with an appropriate action to perform, in this case the sum of the values of `n_crisis` for all the entries that share a date. The output is a dataset containing two columns, named `total` and `df_mov$date`. We rename the second column of the resulting dataset to `date` instead of `df_mov$date`. The final dataset is `df_crisis`.

```r
df_crisis <- df_mov %>%
  group_by(df_mov$date) %>%
  summarise(total = sum(n_crisis)) %>%
  rename("date"="df_mov$date")
```

We apply a very similar process to sum the values of `n_baseline` corresponding to flows from the same day. The resulting dataset is `df_baseline`.

```r
df_baseline <- df_mov %>%
  group_by(df_mov$date) %>%
  summarise(total = sum(n_baseline)) %>%
  rename("date"="df_mov$date")
```

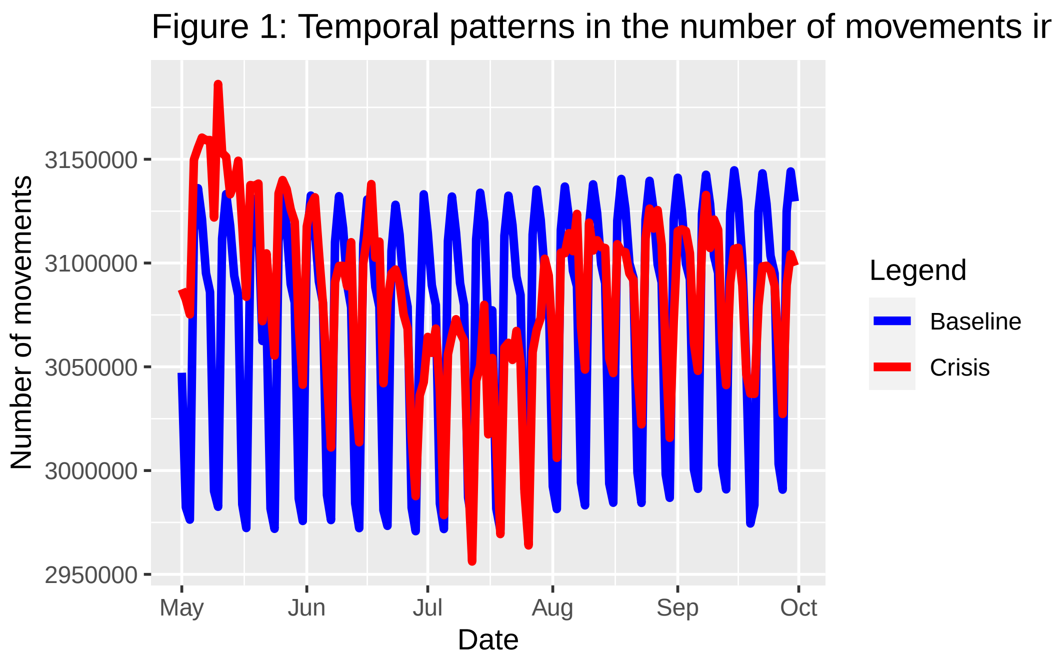
1.3 Internal movements at the nation-wide scale

With `df_crisis` and `df_baseline`, we are ready to create our first figure. The goal is to show the daily evolution of internal movements at the nation-wide scale in Chile during the five-month period starting in May 2020. Therefore, the figure will include information about the date on the x-axis and the number of movements on the y-axis. Furthermore, the figures should allow us to compare the trends in the period of interest with the baseline levels of internal mobility. Hence, our plot will include two components: a line showing the evolution of internal movements from May to September 2020 and a line representing the baseline.

To generate the figure, we use the `ggplot2` package. First, `ggplot()` is used to construct the initial plot object, and is followed by a plus sign (+) to add components to the plot.

Note that we add a legend to specify the component of the plot that refers to the "crisis" or pandemic period and the component that refers to the baseline.

```
ggplot() +
  geom_line(mapping = aes(x = date, y = total, color="Baseline"),
            data = df_baseline, linewidth = 1.5) +
  geom_line(mapping = aes(x = date, y = total, color = "Crisis"),
            data = df_crisis, linewidth = 1.5) +
  scale_color_manual(values = c( "Crisis" = "red", "Baseline" = "blue"),
                     labels = c("Baseline", "Crisis")) +
  labs(color = "Legend",
       title = "Figure 1: Temporal patterns in the number of movements in Chile",
       x = "Date",
       y = "Number of movements")
```

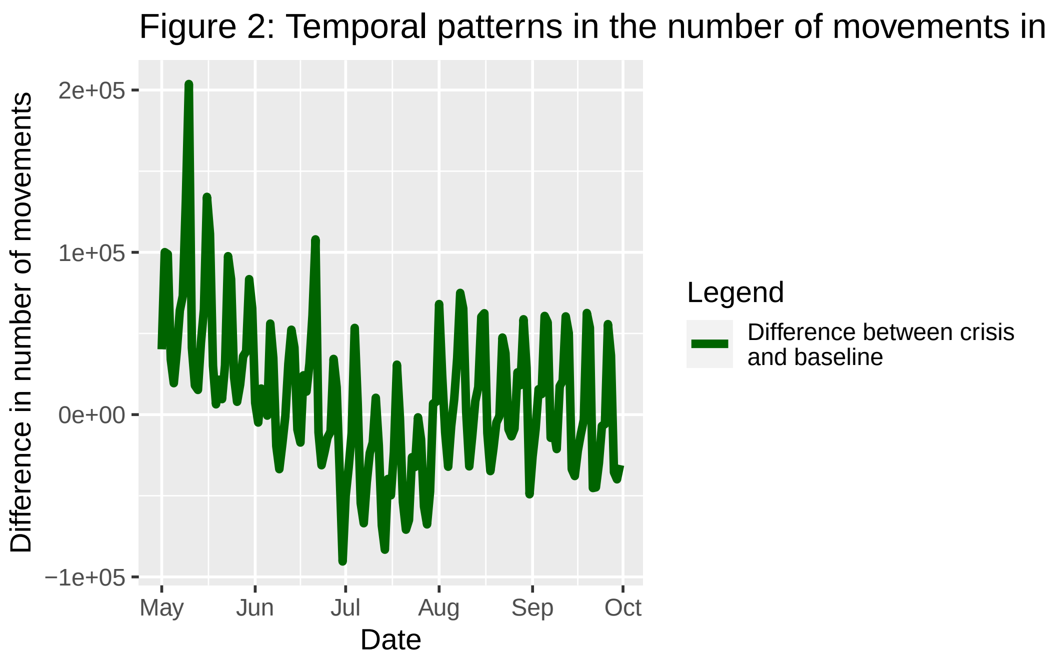


We can also plot the difference between the levels of internal migration during the pandemic period and the baseline. First we create a new dataset `df_difference` with a column named `difference` as follows:

```
df_difference <- df_crisis
df_difference$difference <- df_crisis$total - df_baseline$total
```

Then we generate a plot following a procedure as above:

```r
ggplot() +
  geom_line(mapping = aes(x = date, y = difference, color = "Difference"),
            data = df_difference, linewidth = 1.5) +
  scale_color_manual(values = c("Difference" = "darkgreen"),
            labels = c("Difference between crisis \nand baseline")) +
  labs(color = "Legend",
       title = "Figure 2: Temporal patterns in the number of movements in Chile",
       x = "Date",
       y = "Difference in number of movements")
```



Question 1

The patterns during the crisis period, display an overall trend and some shorter-term oscillations. What do these oscillations correspond to?

Question 2

Does the observed overall trend in the 5-month period starting in May 2020 agree with what you expected? You may answer this question based on your knowledge on the evolution of the COVID-19 pandemic in Chile.

Question 3

Why do you think the number of movements is generally higher during the crisis period than during the baseline period?

1.4 Internal movements involving Santiago province

Up to this point, we have focused on the temporal trends of internal mobility at the nationwide scale. Thanks to the fine level of detail in our original datasets, we can focus the analysis on more specific areas or regions. In the remainder of this sections, we will analyse movements that involve Santiago, Chile's capital city. As we pointed out in section Section 1.2, there are several columns in the dataset `df_mov` that contain information about the start and end locations of the movements. The spatial variability of internal population flows will be explored in more detail in the next section of the workshop. But for now, we will focus on the columns names `start_polygon_name` and `end_polygon_name`, which contain the name of the province where a flow starts or ends respectively.

1.4.1 Internal movements to Santiago

Hence, to obtain only the movements that have the province of Santiago as their destination, we need to filter the data set `df_mov` so that it only contains rows where the value of `end_polygon_name` is Santiago. In R, this can be done very simply by running the lines of code below. The first line, keeps only the flows that start outside of Santiago and stores the result in the `df_mov_to` dataset. The second line, keeps only the flows that end in Santiago and stores the result in the `df_mov_to` dataset. We call the resulting data set `df_move_to`.

```r
df_mov_to <- filter(df_mov, start_polygon_name != 'Santiago')
df_mov_to <- filter(df_mov_to, end_polygon_name == 'Santiago')
```

Then, like before, we create a new column for `df_mov_to` containing only information about the date (and not the time) in year-month-day format:

```r
df_mov_to$date <- as.Date(df_mov_to$date_time, format = "%Y-%m-%d")
```

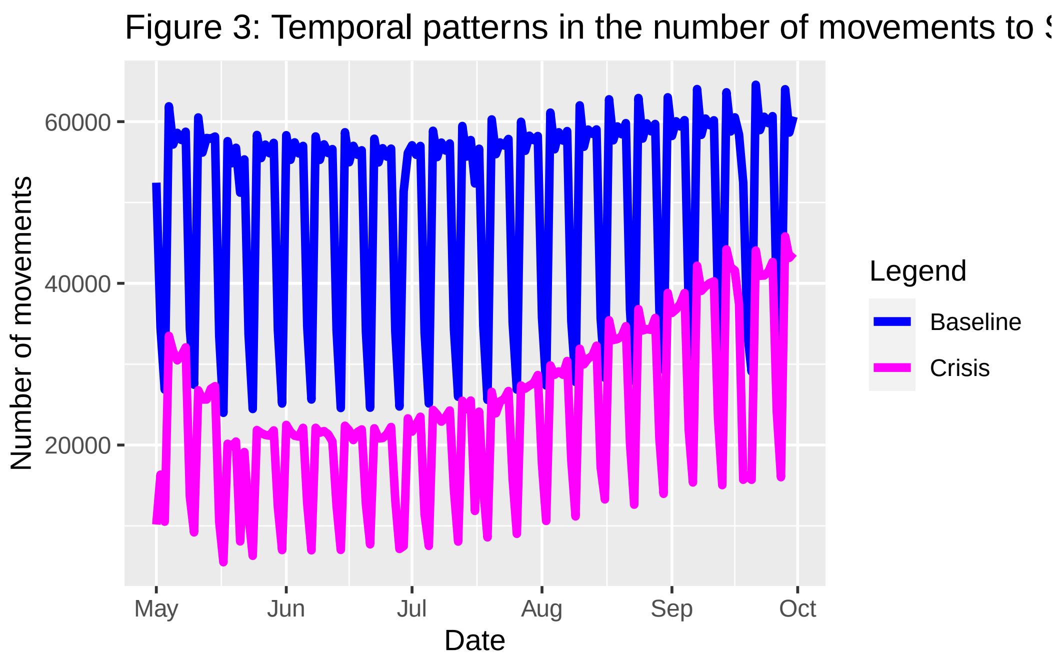
and we follow this action by grouping the data by date. We generate the datasets `df_to_crisis` and `df_to_baseline`, which contain the sum of the values of `n_crisis` and `n_baseline` for flows that happen on the same day:

```r
df_to_crisis <- df_mov_to %>%
  group_by(df_mov_to$date) %>%
  summarise(total = sum(n_crisis)) %>%
  rename("date"="df_mov_to$date") %>%
  ungroup()
```

```r
df_to_baseline <- df_mov_to %>%
  group_by(df_mov_to$date) %>%
  summarise(total = sum(n_baseline)) %>%
  rename("date"="df_mov_to$date") %>%
  ungroup()
```

With all these elements, we can generate a figure showing the trends in the number of movements from any province outside of Santiago to Santiago, both during the 5-month crisis period during the pandemic and during the baseline period:

```r
ggplot() +
  geom_line(mapping = aes(x = date, y = total, color="Baseline"),
            data = df_to_baseline, linewidth = 1.5) +
  geom_line(mapping = aes(x = date, y = total, color = "Crisis"),
            data = df_to_crisis, linewidth = 1.5) +
  scale_color_manual(values = c( "Crisis" = "magenta", "Baseline" = "blue"),
                     labels = c("Baseline", "Crisis")) +
  labs(color = "Legend",
       title = "Figure 3: Temporal patterns in the number of movements to Santiago",
       x = "Date",
       y = "Number of movements")
```



1.4.2 Internal movements from Santiago

An analogous analysis can be carried out for movements with origin in the province of Santiago and destination elsewhere. To do this, we follow exactly the same steps as before but modifying the way in which we filter `df_mov`, and replacing the names of new variables where appropriate (e.g. `df_mov_from` instead of `df_mov_to`):

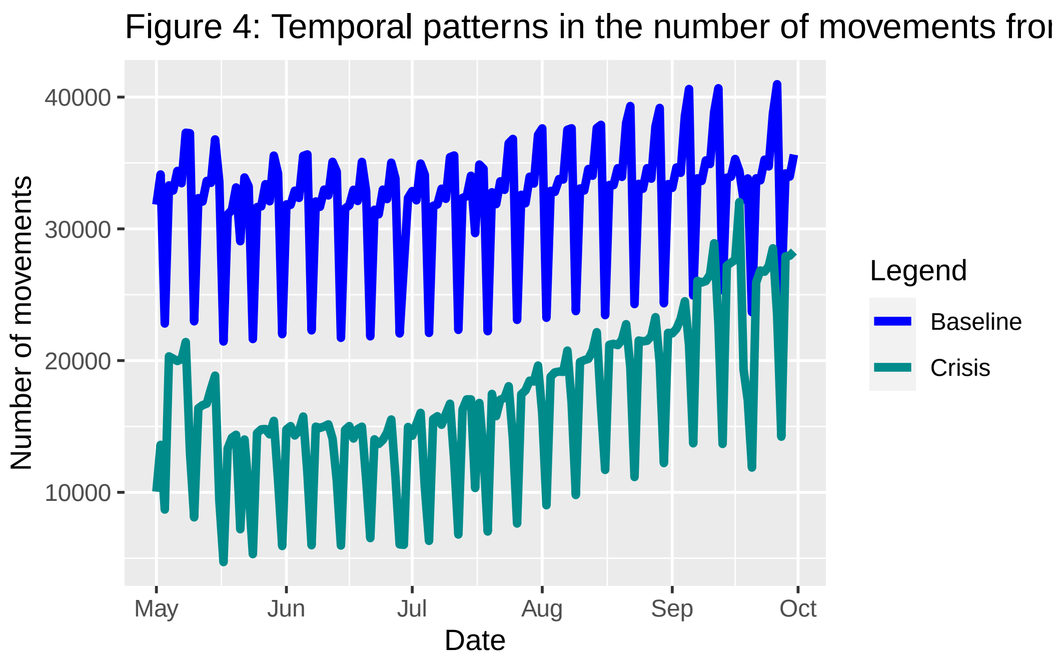
```
df_mov_from <- filter(df_mov, start_polygon_name == 'Santiago')
df_mov_from <- filter(df_mov_from, end_polygon_name != 'Santiago')
```

```
df_mov_from$date <- as.Date(df_mov_from$date_time, format = "%Y-%m-%d")
```

```
df_from_crisis <- df_mov_from %>%
  group_by(df_mov_from$date) %>%
  summarise(total = sum(n_crisis)) %>%
  rename("date"="df_mov_from$date") %>%
  ungroup()
```

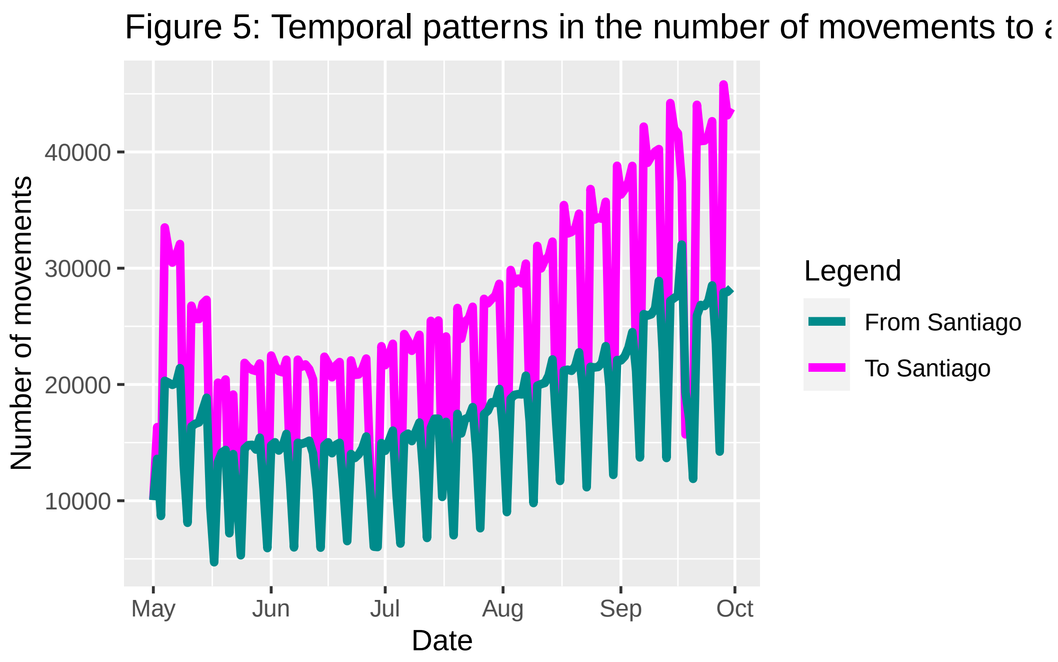
```
df_from_baseline <- df_mov_from %>%
  group_by(df_mov_from$date) %>%
  summarise(total = sum(n_baseline)) %>%
  rename("date"="df_mov_from$date") %>%
  ungroup()
```

```
ggplot() +
  geom_line(mapping = aes(x = date, y = total, color="Baseline"),
            data = df_from_baseline, linewidth = 1.5) +
  geom_line(mapping = aes(x = date, y = total, color = "Crisis"),
            data = df_from_crisis, linewidth = 1.5) +
  scale_color_manual(values = c( "Crisis" = "darkcyan", "Baseline" = "blue"),
                     labels = c("Baseline", "Crisis")) +
  labs(color = "Legend",
       title = "Figure 4: Temporal patterns in the number of movements from Santiago",
       x = "Date",
       y = "Number of movements")
```



We can also compare the movements to and from Santiago province, this time omitting the baseline trends for clarity:

```
ggplot() +
  geom_line(mapping = aes(x = date, y = total, color="To"),
            data = df_to_crisis, size = 1.5) +
  geom_line(mapping = aes(x = date, y = total, color = "From"),
            data = df_from_crisis, size = 1.5) +
  scale_color_manual(values = c( "To" = "magenta", "From" = "darkcyan"),
                     labels = c("From Santiago", "To Santiago")) +
  labs(color = "Legend",
       title = "Figure 5: Temporal patterns in the number of movements to and from Santiag
       x = "Date",
       y = "Number of movements")
```



Question 4

Does the overall trend in the internal movements involving Santiago province agree with your intuition?

Question 5

Can you observe any differences in the shorter-term oscillations corresponding to movements to and from Santiago province? To answer this question, it might be helpful to complete the Exercise below.

Exercise 1

Create a plot analogous to Figure 5, but comparing the baseline movements instead of the movements during the crisis period.

1.4.3 Internal movements between Santiago and Valparaíso

It is possible to look at movements beetween a specified origin and destination. Here, we focus on the flow of people between Santiago and a popular destination province for Santiaguinos, Valparaíso. To explore the movements from Santiago to Valparaíso, you can run the code below:

```r
df_mov_between1 <- filter(df_mov, start_polygon_name == 'Santiago')
df_mov_between1 <- filter(df_mov_between1, end_polygon_name == 'Valparaíso')
```

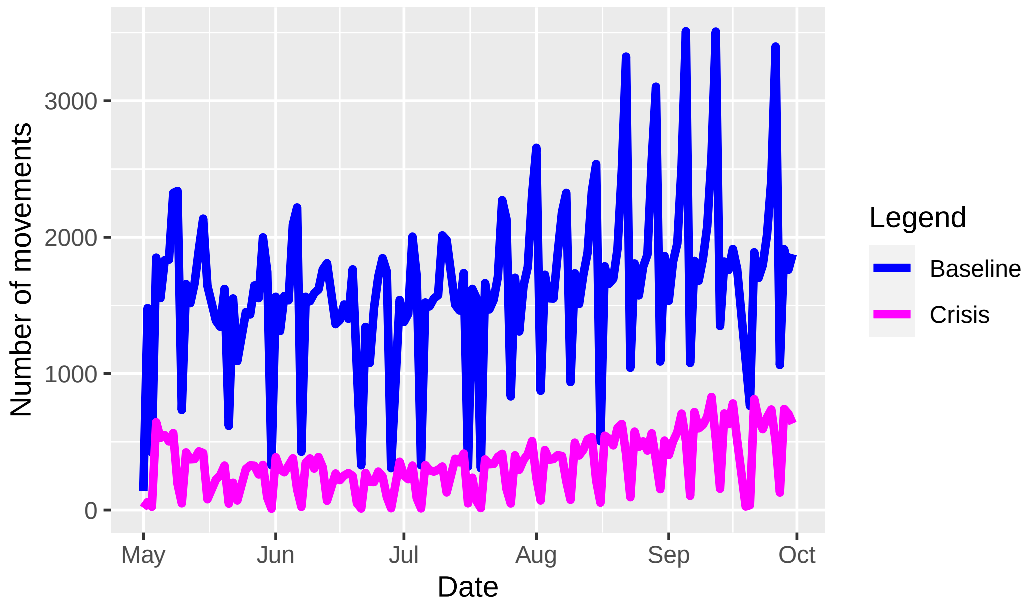
```r
df_mov_between1$date <- as.Date(df_mov_between1$date_time, format = "%Y-%m-%d")
```

```r
df_between1_crisis <- df_mov_between1 %>%
  group_by(df_mov_between1$date) %>%
  summarise(total = sum(n_crisis)) %>%
  rename("date"="df_mov_between1$date") %>%
  ungroup()
```

```r
df_between1_baseline <- df_mov_between1 %>%
  group_by(df_mov_between1$date) %>%
  summarise(total = sum(n_baseline)) %>%
  rename("date"="df_mov_between1$date") %>%
  ungroup()
```

```r
ggplot() +
  geom_line(mapping = aes(x = date, y = total, color="Baseline"),
            data = df_between1_baseline, size = 1.5) +
  geom_line(mapping = aes(x = date, y = total, color = "Crisis"),
            data = df_between1_crisis, size = 1.5) +
  scale_color_manual(values = c( "Crisis" = "magenta", "Baseline" = "blue"),
                     labels = c("Baseline", "Crisis")) +
  labs(color = "Legend",
       title = "Figure 6: Temporal patterns in the number of movements from Santiago to Va
       x = "Date",
        y = "Number of movements")
```

Figure 6: Temporal patterns in the number of movements fron



Question 6

Does the overall trend in the internal movements from Santiago to Valparaíso in this time period agree with your intuition?

Exercise 2

Create a plot analogous to Figure 6, but comparing the movements from Valparaíso to Santiago during the crisis and baseline period.

Exercise 3

Compare the trends in movements from Santiago to Valparaíso and from Valparaíso to Santiago. Are the overall trends similar? Are there differentces in the shorter-term fluctuations?

1.5 Advanced: tracked internal movements in relation to the active population of Facebook users

So far, we have looked at the 5-month period starting in May 2020 and spanning throughout the Chilean winter. But, as mentioned at the beginning of the workbook, Facebook Meta collected data for Chile in the two-year period spanning from the end of March of 2020 to the end of March 2022. As we saw in Section 1.2, the data is organised in files by month, and we can load them individually. However, since we are going to look at many months, we can automate the data-reading process. This is done by first, specifying the path to the folder where the movement data files are stored, and then, using a for loop to read each of the files in the folder and bind them together. Getting a full understanding of the code below is out of the scope of this workshop, but we have included comments starting with the (#) symbol to indicate what each line of code does in case you are curious.

```
# Get a list of file paths in the folder
folder_path <- "./data/fb/movement_grid/"
file_paths <- list.files(path = folder_path, pattern = "\\.rds$", full.names = TRUE)

# Create an empty list to store the data frames
dfs <- list()

# Loop through the file paths
for (file_path in file_paths) {
  # Read each file as a data frame
  df <- readRDS(file_path)
  # Store the data frame in the list
  dfs <- append(dfs, list(df))
}

# Bind all the data frames into a single data frame
df_mov_full <- bind_rows(dfs)
```

In the previous sections, we have focused on analysing movement data during the pandemic and comparing it to a baseline period. While some insights can be gained by looking only at this data, it is important to consider the number of active Facebook users. The reason for this can be illustrated through the following examples. Suppose that the number of Facebook users drops to zero, then no movements would be recorded. However, this does not mean there are no movements taking place. Conversely, suppose Facebook becomes very popular and the whole population starts using it, then more movements would probably be recorded but this does not mean that people started moving more. Hence, it can be very useful to look at the number of movements per active user and analyse how this quantity evolves both during the pandemic and the baseline.

Below, we load data for the population of active Facebook users:

```r
# Get a list of file paths in the folder
folder_path <- "data/fb/population_grid/"
file_paths <- list.files(path = folder_path, pattern = "\\.rds$", full.names = TRUE)

# Create an empty list to store the data frames
dfs <- list()

# Loop through the file paths
for (file_path in file_paths) {
  # Read each file as a data frame
  df <- readRDS(file_path)

  # Store the data frame in the list
  dfs <- append(dfs, list(df))
}

# Bind all the data frames into a single data frame
df_mov_full_pop <- bind_rows(dfs)
```

1.5.1 Pre-processing the data

Like the movement datasets, the Facebook population data is aggregated into 8-hour time windows but here, we are only interested in the variation at the daily level. Therefore, we group by date the number of movements and the number of active users during the crisis period.

```r
df_mov_full$date <- as.Date(df_mov_full$date_time, format = "%Y-%m-%d")
```

```r
df_mov_full_pop$date <- as.Date(df_mov_full_pop$date_time, format = "%Y-%m-%d")
```

```r
df_mov_full_crisis <- df_mov_full %>%
  group_by(df_mov_full$date) %>%
  summarise(n_crisis = sum(n_crisis)) %>%
  rename("date"="df_mov_full$date") %>%
  ungroup()
```

```
df_mov_full_pop_crisis <- df_mov_full_pop %>%
  group_by(df_mov_full_pop$date) %>%
  summarise(pop_n_crisis = sum(n_crisis)) %>%
  rename("date"="df_mov_full_pop$date") %>%
  ungroup()
```

Once the grouping by date is done, we merge the resulting datasets for movements and population of active Facebook users according to the `date` column:

```
df_full_crisis <- merge(df_mov_full_crisis, df_mov_full_pop_crisis, by = "date", all.x = T
```

The original data for the population of active Facebook users is missing some dates, so we estimate the missing data using Multivariate Imputation by Chained Equations (with the mice package).

```
# Create a mice object with only the column you want to impute
df_to_complete <- mice(df_full_crisis[, -which(names(df_full_crisis) == "n_crisis")])
```

```
 iter imp variable
  1   1  pop_n_crisis
  1   2  pop_n_crisis
  1   3  pop_n_crisis
  1   4  pop_n_crisis
  1   5  pop_n_crisis
  2   1  pop_n_crisis
  2   2  pop_n_crisis
  2   3  pop_n_crisis
  2   4  pop_n_crisis
  2   5  pop_n_crisis
  3   1  pop_n_crisis
  3   2  pop_n_crisis
  3   3  pop_n_crisis
  3   4  pop_n_crisis
  3   5  pop_n_crisis
  4   1  pop_n_crisis
  4   2  pop_n_crisis
  4   3  pop_n_crisis
  4   4  pop_n_crisis
  4   5  pop_n_crisis
  5   1  pop_n_crisis
  5   2  pop_n_crisis
  5   3  pop_n_crisis
  5   4  pop_n_crisis
  5   5  pop_n_crisis
```

```
# Perform the imputation
df_complete_crisis <- complete(df_to_complete)
df_full_crisis$pop_n_crisis <- df_complete_crisis$pop_n_crisis
```

Similarly, we group by date the number of movements and the number of active users during the baseline. Then, we merge the resulting datasets for movements and population of active Facebook users according to the `date` column and finally, we estimate any missing values using Multivariate Imputation by Chained Equations (with the mice package).

```
df_mov_full_baseline <- df_mov_full %>%
  group_by(df_mov_full$date) %>%
  summarise(n_baseline = sum(n_baseline)) %>%
  rename("date"="df_mov_full$date") %>%
  ungroup()
```

```
df_mov_full_pop_baseline <- df_mov_full_pop %>%
  group_by(df_mov_full_pop$date) %>%
  summarise(pop_n_baseline = sum(n_baseline)) %>%
  rename("date"="df_mov_full_pop$date") %>%
  ungroup()
```

```
df_full_baseline <- merge(df_mov_full_baseline, df_mov_full_pop_baseline, by = "date", all
```

```
# Create a mice object with only the column you want to impute
df_to_complete <- mice(df_full_baseline[, -which(names(df_full_baseline) == "n_baseline")]
```

```
iter imp variable
 1   1  pop_n_baseline
 1   2  pop_n_baseline
 1   3  pop_n_baseline
 1   4  pop_n_baseline
 1   5  pop_n_baseline
 2   1  pop_n_baseline
 2   2  pop_n_baseline
 2   3  pop_n_baseline
```

```
2   4  pop_n_baseline
2   5  pop_n_baseline
3   1  pop_n_baseline
3   2  pop_n_baseline
3   3  pop_n_baseline
3   4  pop_n_baseline
3   5  pop_n_baseline
4   1  pop_n_baseline
4   2  pop_n_baseline
4   3  pop_n_baseline
4   4  pop_n_baseline
4   5  pop_n_baseline
5   1  pop_n_baseline
5   2  pop_n_baseline
5   3  pop_n_baseline
5   4  pop_n_baseline
5   5  pop_n_baseline
```

```r
# Perform the imputation
df_complete_baseline <- complete(df_to_complete)
df_full_baseline$pop_n_baseline <- df_complete_baseline$pop_n_baseline
```

The merged datasets for the crisis period and the baseline can be merged according to the `date` column:

```r
df_full <- merge(df_full_crisis, df_full_baseline, by = "date", all.x = TRUE)
```

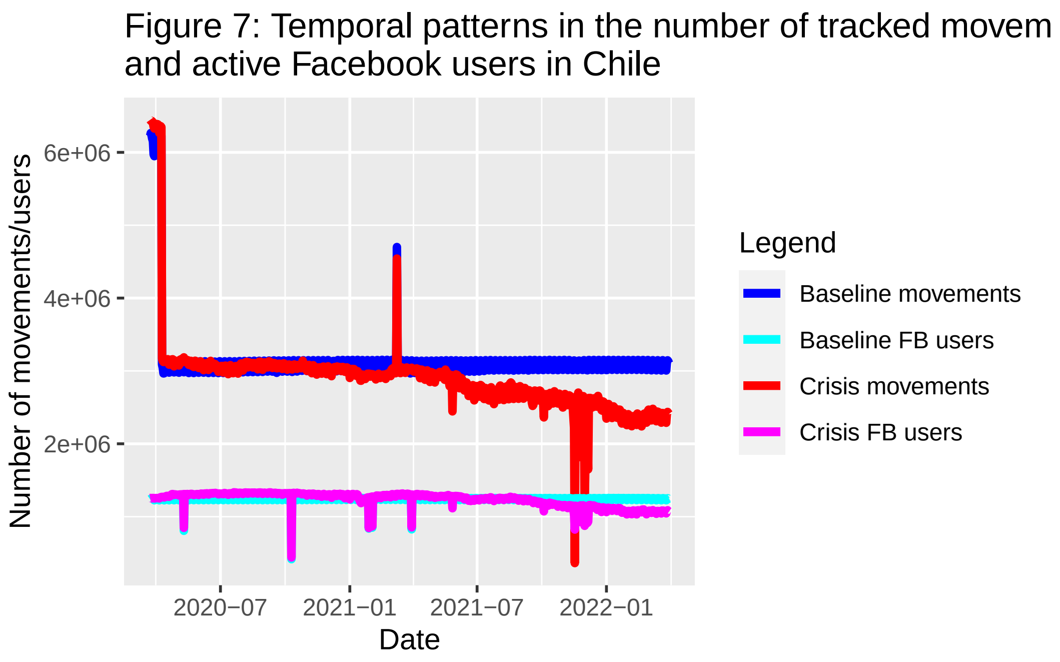
To obtain the number of movements per active Facebook user on a given date, we can divide the number of movements on that date by the total number of active Facebook users on the same date. This can be done for both the crisis and baseline periods and the results can be stored in new columns. Furthermore, we compute the difference in the number of movements per active Facebook user between the crisis and baseline periods and store it in the `difference_div` column:

```r
df_full$n_crisis_div <- df_full$n_crisis/df_full$pop_n_crisis

df_full$n_baseline_div <- df_full$n_baseline/df_full$pop_n_baseline

df_full$n_difference_div <- df_full$n_crisis_div - df_full$n_baseline_div
```

1.5.2 Analysisng the results

Below we plot the baseline number of tracked movements and Facebook active users as well as the same quantities for the crisis period:

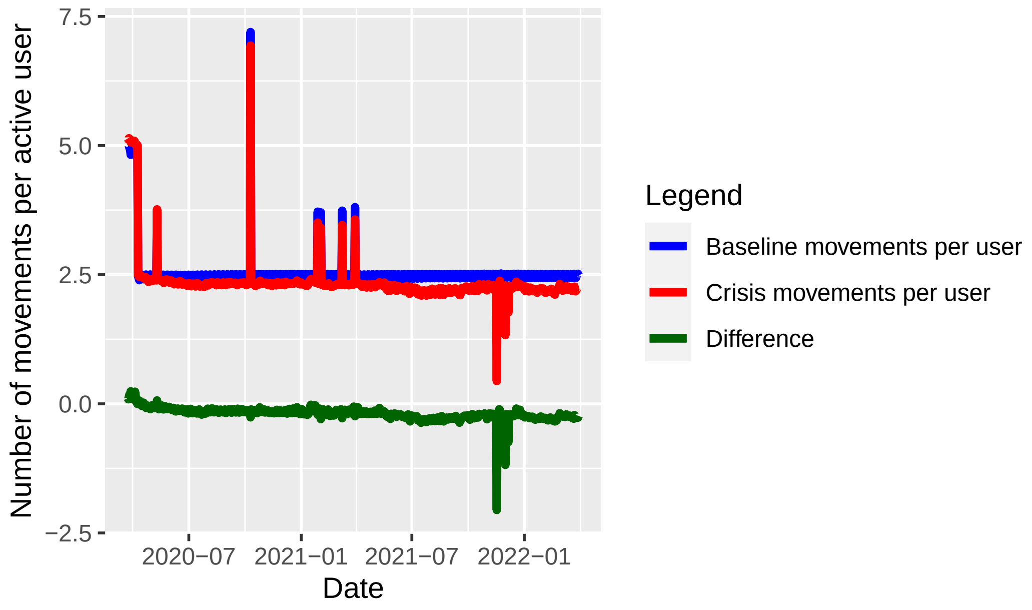
```
ggplot() +

  geom_line(mapping = aes(x = date, y = n_baseline, color="Baseline"),
              data = df_full, size = 1.5) +
  geom_line(mapping = aes(x = date, y = n_crisis, color = "Crisis"),
            data = df_full, size = 1.5)  +
  geom_line(mapping = aes(x = date, y = pop_n_baseline, color="Baseline FB users"),
              data = df_full, size = 1.5) +
  geom_line(mapping = aes(x = date, y = pop_n_crisis, color = "Crisis FB users"),
            data = df_full, size = 1.5)  +
  scale_color_manual(values = c( "Crisis" = "red", "Baseline" = "blue", "Crisis FB users"
  labs(color = "Legend", title = "Figure 7: Temporal patterns in the number of tracked mov
        x = "Date",
        y = "Number of movements/users")
```



Disregarding the spikes which are likely due to anomalies in the data collection process applied by Facebook, we observe that, generally speaking, both the baseline number of movements and the number of active Facebook users remain constant through the plotted period. In the case of the number of movements and the number of active Facebook users during the pandemic, they both experience a decline. But, is the decline observed in the number of movements a result of the decline in the number of active Facebook users? To answer this question, we can plot the number of tracked movements per active Facebook user, both for the baseline and the pandemic period. We can also plot the difference between these two quantities:

```r
ggplot() +
  geom_line(mapping = aes(x = date, y = n_baseline_div, color="Baseline movements per user
             data = df_full, size = 1.5) +
  geom_line(mapping = aes(x = date, y = n_crisis_div, color = "Crisis movements per user")
           data = df_full, size = 1.5)  +
  geom_line(mapping = aes(x = date, y = n_difference_div, color = "Difference"),
           data = df_full, size = 1.5)  +
  scale_color_manual(values = c( "Crisis movements per user" = "red", "Baseline movements
  expand_limits(y = 0) +
  labs(color = "Legend", title = "Figure 8: Temporal patterns in the number of tracked mov
       x = "Date",
       y = "Number of movements per active user")
```

Figure 8: Temporal patterns in the number of tracked moveme



This last figure shows that since the start of the pandemic, the number of movements per active Facebook user has been consistently lower than during the baseline period. However, the difference in trends between the crisis and basline periods is not as large as it might have been suggested by looking at the raw number of movements without taking into account the active population of users (e.g. the line corresponding to the number of movements during the crisis period from Figure 7).

2 Spatial patterns

2.1 Aims

2.2 Dependencies

```
# data wrangling
library(tidyverse)
#library(fs)
#library(here)

# spatial data wrangling
library(sf)
library(mapdeck)

# data visualisation
library(viridis)
#library(viridisLite)
library(ggthemes)
library(patchwork)
library(showtext)
library(gganimate)
#library(gifski)
#library(ggnewscale)
```

2.3 Data

Read and describe Meta-Facebook mobility data

```
df20 <- readRDS("./data/fb/movement_adm/2020_04.rds") %>%
  dplyr::filter(country == "CL")
str(df20)
```

```
spc_tbl_ [29,491 x 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ GEOMETRY                    : chr [1:29491] "LINESTRING (-69.96872527689874 -23.40113045
 $ date_time                   : chr [1:29491] "2020-04-01 00:00" "2020-04-01 00:00" "2020-0
 $ start_polygon_id            : chr [1:29491] "60845" "60862" "60845" "60862" ...
 $ start_polygon_name          : chr [1:29491] "Antofagasta" "Cardenal Caro" "Antofagasta" "
 $ end_polygon_id              : chr [1:29491] "60847" "60890" "60846" "60863" ...
 $ end_polygon_name            : chr [1:29491] "Tocopilla" "Melipilla" "El Loa" "Colchagua"
 $ length_km                   : num [1:29491] 139.8 24.1 100.3 24.2 115.5 ...
 $ tile_size                   : num [1:29491] 11 11 11 11 11 11 11 11 11 11 ...
 $ country                     : chr [1:29491] "CL" "CL" "CL" "CL" ...
 $ level                       : chr [1:29491] "LEVEL3" "LEVEL3" "LEVEL3" "LEVEL3" ...
 $ n_crisis                    : num [1:29491] 79 18 320 71 NA 369 NA 20 11 NA ...
 $ n_baseline                  : num [1:29491] 59.5 25.5 671 132.8 NA ...
 $ n_difference                : num [1:29491] 19.5 -7.5 -351 -61.8 NA ...
 $ percent_change              : num [1:29491] 32.2 -28.3 -52.2 -46.2 166.7 ...
 $ is_statistically_significant: num [1:29491] 0 0 0 0 0 0 0 0 0 0 ...
 $ z_score                     : num [1:29491] 2.015 -0.684 -4 -1.261 1.5 ...
 $ start_lat                   : num [1:29491] -24.3 -34.3 -24.3 -34.3 -32.6 ...
 $ start_lon                   : num [1:29491] -69.6 -71.8 -69.6 -71.8 -70.7 ...
 $ end_lat                     : num [1:29491] -22.1 -33.8 -22.8 -34.7 -33.7 ...
 $ end_lon                     : num [1:29491] -69.6 -71.2 -68.2 -71.1 -70.9 ...
 - attr(*, "spec")=
  .. cols(
  ..   GEOMETRY = col_character(),
  ..   date_time = col_character(),
  ..   start_polygon_id = col_character(),
  ..   start_polygon_name = col_character(),
  ..   end_polygon_id = col_character(),
  ..   end_polygon_name = col_character(),
  ..   length_km = col_double(),
  ..   tile_size = col_double(),
  ..   country = col_character(),
  ..   level = col_character(),
  ..   n_crisis = col_double(),
  ..   n_baseline = col_double(),
  ..   n_difference = col_double(),
  ..   percent_change = col_double(),
  ..   is_statistically_significant = col_double(),
  ..   z_score = col_double(),
  ..   start_lat = col_double(),
  ..   start_lon = col_double(),
  ..   end_lat = col_double(),
  ..   end_lon = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
unique_origins <- unique(df20$start_polygon_name)
unique_destinations <- unique(df20$end_polygon_name)
```

2.3.1 Bing tiles

Read and describe Bing tile grids and how they can be created - reference to our work.

```
#bing_grid <- read_sf()
```

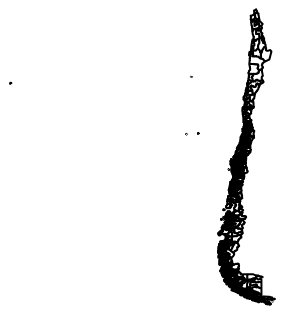
2.3.2 Administrative areas

Read and describe administrative areas. Explain spatial data frames: geometry, projection, etc.

```
shp <- read_sf("./data/shp/gadm41_CHL_3.shp") %>%
  st_simplify(preserveTopology =T,
              dTolerance = 1000) %>%  # 1km
  sf::st_make_valid() %>%
  dplyr::select( -c(
    NL_NAME_1, NL_NAME_2, VARNAME_3, NL_NAME_3, CC_3, HASC_3
  ) )

shp
```

```
Simple feature collection with 345 features and 10 fields (with 1 geometry empty)
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -109.4488 ymin: -55.97871 xmax: -66.41821 ymax: -17.49952
Geodetic CRS:  WGS 84
# A tibble: 345 x 11
   GID_3       GID_0 COUNTRY GID_1   NAME_1 GID_2 NAME_2 NAME_3 TYPE_3 ENGTYPE_3
   <chr>       <chr> <chr>   <chr>   <chr>  <chr> <chr>  <chr>  <chr>  <chr>
 1 CHL.2.1.1_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Antof~ Antof~ Comuna Municipa~
 2 CHL.2.1.2_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Antof~ Mejil~ Comuna Municipa~
 3 CHL.2.1.3_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Antof~ Sierr~ Comuna Municipa~
 4 CHL.2.1.4_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Antof~ Taltal Comuna Municipa~
 5 CHL.2.2.1_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ El Loa Calama Comuna Municipa~
 6 CHL.2.2.2_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ El Loa Ollag~ Comuna Municipa~
 7 CHL.2.2.3_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ El Loa San P~ Comuna Municipa~
 8 CHL.2.3.1_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Tocop~ María~ Comuna Municipa~
 9 CHL.2.3.2_1 CHL   Chile   CHL.2_1 Antof~ CHL.~ Tocop~ Tocop~ Comuna Municipa~
10 CHL.3.1.1_1 CHL   Chile   CHL.3_1 Arauc~ CHL.~ Cautín Carah~ Comuna Municipa~
# i 335 more rows
# i 1 more variable: geometry <POLYGON [°]>
```

```
plot(shp$geometry)
```



2.4 Spatial indicators of human mobility

2.4.1 Origin-based indicators

This measure is in relation to a baseline - percentage change and flow

```
origin_df <- df20 %>%
  group_by(start_polygon_name) %>%
  dplyr::summarise(
    mean_perchange = mean(percent_change, na.rm = T),
    mean_flows = mean(n_difference, na.rm = T),
    mean_outflow = mean(n_crisis, na.rm = T),
    sum_flows = sum(n_difference, na.rm = T),
    sum_outflow = sum(n_crisis, na.rm = T)
    ) %>%
  ungroup()

tail(origin_df, 10)
```

```
# A tibble: 10 x 6
   start_polygon_name mean_perchange mean_flows mean_outflow sum_flows
   <chr>                       <dbl>      <dbl>        <dbl>     <dbl>
 1 Santiago                    -67.6      914.        30810.  1593286.
 2 Talagante                   -27.3      368.         3294.   212183.
 3 Talca                       -47.2      111.         5641.    62325.
 4 Tamarugal                    15.1     -275.         1380.   -62324.
 5 Tierra del Fuego            -22.9       11.6         832.     1109.
 6 Tocopilla                   119.        40.0         368.    10228.
 7 Valdivia                    -63.9     -883.         6014.  -325912.
 8 Valparaíso                  -57.6     -453.        11183.  -374287.
 9 Ñuble                       -54.1     -164.         7629.   -70690.
10 Última Esperanza            -14.8     -162.          705.   -14771.
# i 1 more variable: sum_outflow <dbl>
```

2.4.2 Destination-based indicators

```
destination_df <- df20 %>%
  group_by(end_polygon_name) %>%
  dplyr::summarise(
    mean_perchange = mean(percent_change, na.rm = T),
    mean_flows = mean(n_difference, na.rm = T),
    mean_outflow = mean(n_crisis, na.rm = T),
    sum_flows = sum(n_difference, na.rm = T),
    sum_outflow = sum(n_crisis, na.rm = T)
    ) %>%
  ungroup()

tail(destination_df, 10)
```

```
# A tibble: 10 x 6
   end_polygon_name mean_perchange mean_flows mean_outflow sum_flows sum_outflow
   <chr>                     <dbl>      <dbl>        <dbl>     <dbl>       <dbl>
 1 Santiago                  -69.0      943.        32307.  1566780.    53693876
 2 Talagante                 -24.2      371.         3324.   211701.     1897958
 3 Talca                     -46.7      115.         5682.    64266.     3170655
 4 Tamarugal                  18.8     -261.         1314.   -62423.      313993
 5 Tierra del Fuego          -21.7       11.5         823.     1116.       79867
 6 Tocopilla                  53.1       39.3         345.    10766.       94540
 7 Valdivia                  -60.6     -887.         6079.  -323796.     2218811
 8 Valparaíso                -61.3     -446.        11374.  -362437.     9235667
 9 Ñuble                      -54.5     -158.          7229.   -72077.     3296434
10 Última Esperanza           -17.7     -164.           712.   -14750.       64118
```

2.4.3 Netflows

2.5 Mapping

2.5.1 Choropleths

2.5.2 Interactive mapping

2.5.3 Flow mapping

2.6 Common concepts

There are several concepts relevant to understanding the data sets. First of all, we construct maps using two different methods of identifying locations: tiles and administrative polygons.

The Bing Maps Tile System defines a series of grids at different resolution levels over a rectangular projection of the world (Schwartz 2018). Each level is constructed by dividing the previous level into fourths. We typically use Bing tile levels 13 through 16, where level 13 results in tiles that are about 4.9 x 4.9 km at the Equator. The other method we use for identifying a location is administrative polygons, which define the political and geographic boundaries of countries, states, provinces, counties, cities, and more.

When generating a map for a crisis event, we specify a rectangular bounding box around the most directly affected area. The different map calculations, described in the following sections, are done relative to this region, and, for most of the maps, only data within this region is included. Most of the map types are based on counting events that occur within a time interval, which is frequently 8 or 24 hours. The time interval determines what data is included in a calculation as well as the minimum frequency with which new maps are generated.