Exploring mobility dynamics in FVG - part 1: data preparation

Table of contents

1 Context and objectives

The dataset consists of information regarding mobility within the Region and includes the followinf information: origin and destination, number of journeys, date, time and type of journey. Specifically, each entry in the dataset represents a group of journeys undertaken by individuals.

The dataset also includes date and time feature: journeys are grouped by day and into blocks of three hours. Additionally, the 'type' attribute distinguishes between residents and tourists, and for residents between inbound and outbound journeys.

The purpose of this notebook is data preparation and exploration of the dataset 'mobilityFVG'.

The main output is a tidy dataset in `.csv` format.

2 Data preparation

The dataset is provided as multiple Excel files (.xlsx format), and requires cleaning for analysis. Tasks include:

- Consolidating multiple xlsx files into a single .csv file to ensure data consistency and facilitates streamlined analysis
- Renaming variables to English
- Removing special characters and whitespaces
- Derive categorical variables from the 'type' column, simplifying them into more efficient categories such as 'inbound' and 'outbound'
- Set appropriate date format, and calculate weekday (1 = Monday)
- Eliminating unnecessary columns.

The R code utilizes the 'readxl' package to import data from Excel files and the 'tidyverse' package for efficient data manipulation. Specifically, tidyverse' an efficient syntax and verbs like 'mutate', 'rename', and 'select' to clean and preprocess the dataset, enhancing readability and facilitating code debug and reuse.

```
library(tidyverse)
library(readxl)
library(gridExtra)
```

Data sources for this notebook

```
input_files <- c('./data/EXPORT FVG _ 6-10 MARZO 2023.xlsx',
                 './data/EXPORT FVG _ 11-15 MARZO 2023.xlsx',
                 './data/EXPORT FVG _ 1-5 GIUGNO 2023.xlsx',
                 './data/EXPORT FVG _ 6-10 GIUGNO 2023.xlsx')
```

Parameters for data preparation: columns to keep and their new names

```
column_names <- c( "Giorno" ,
                   "Comune di partenza", "Area di censimento di partenza",
                   "Comune di arrivo", "Area di censimento di arrivo" ,
                   "Provenienza", "Tipologia viaggiatore", "Viaggi",
                   "Italiani", "Stranieri",
                   "Età 18-24", "Età 25-34", "Età 34-45",
                   "Età 45-54", "Età 55-64", "Età 65+",
                   "00-03", "03-06", "06-09", "09-12",
```

```r
                    "12-15", "15-18", "18-21", "21-24")

selected_cols <- c( "Giorno",
                    "Comune di partenza", "Comune di arrivo",
                    "Tipologia viaggiatore", "Viaggi")

new_column_names <- c( "day", "origin", "destination", "type", "n")
```

Define a function that reads a single file and processes data:

```r
read_and_clean <- function(df,
                           selected_cols,
                           new_column_names) {
  df <- df %>%
  # remove unnecessary columns
  select(all_of(selected_cols)) %>%

  # rename columns
  setNames(new_column_names) %>%

  #replace spaces with _ in column names
  rename_with(~ gsub(" ", "_", .), everything()) %>%

  #remove special characters such as ò, à ...
  mutate(across(where(is.character), ~ gsub("[^a-zA-Z0-9\\s]", " ", .))) %>%

  #set appropriate format for Day and get weekday
  mutate(day = as.Date(day, format = "%d-%m-%Y")) %>%
  mutate(weekday = as.numeric(format(day, "%u"))) %>%

  # encode direction as inbound or outbound
  mutate(direction = case_when(
    grepl("di ritorno da altro Comune", type) ~ "inbound",
    grepl("diretto in altro Comune", type) ~ "outbound",
    TRUE ~ "--")) %>%

  # encode type of traveller
  mutate(res_trav = case_when(
    grepl("Residente", type) ~ "resident",
    grepl("Viaggiatore", type) ~ "traveller",
    TRUE ~ "--")) %>%
      select(-type)

     return(df)
}
```

apply the function to each file and save in a single `dataframe`

```r
data <- data.frame()
for (filename in input_files){
    print(paste("Reading and processing", filename))
    df <- read_excel(filename)
    if( all(column_names == colnames(df)) ){
        print("Colnames are OK, adding data.")
        data <- rbind(data,
                      read_and_clean(df,
                                     selected_cols,
                                     new_column_names))
    } else {
        print("Column names do not match: skipping this file!")
    }


}
```

```
[1] "Reading and processing ./data/EXPORT FVG _ 6-10 MARZO 2023.xlsx"
[1] "Colnames are OK, adding data."
[1] "Reading and processing ./data/EXPORT FVG _ 11-15 MARZO 2023.xlsx"

New names:
* `21-24` -> `21-24...23`
* `21-24` -> `21-24...24`

[1] "Column names do not match: skipping this file!"
[1] "Reading and processing ./data/EXPORT FVG _ 1-5 GIUGNO 2023.xlsx"
[1] "Column names do not match: skipping this file!"
[1] "Reading and processing ./data/EXPORT FVG _ 6-10 GIUGNO 2023.xlsx"
[1] "Colnames are OK, adding data."
```

```r
glimpse(data, width = 60)
```

```
Rows: 542,276
Columns: 7
$ day         <date> 2023-03-06, 2023-03-06, 2023-03-06, 2~
$ origin      <chr> "Porcia", "Pordenone", "Pordenone", "C~
$ destination <chr> "Pordenone", "Porcia", "Cordenons", "P~
$ n           <dbl> 1876, 1794, 1695, 1667, 1425, 1366, 13~
$ weekday     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ direction   <chr> "outbound", "inbound", "inbound", "out~
$ res_trav    <chr> "resident", "resident", "resident", "r~
```

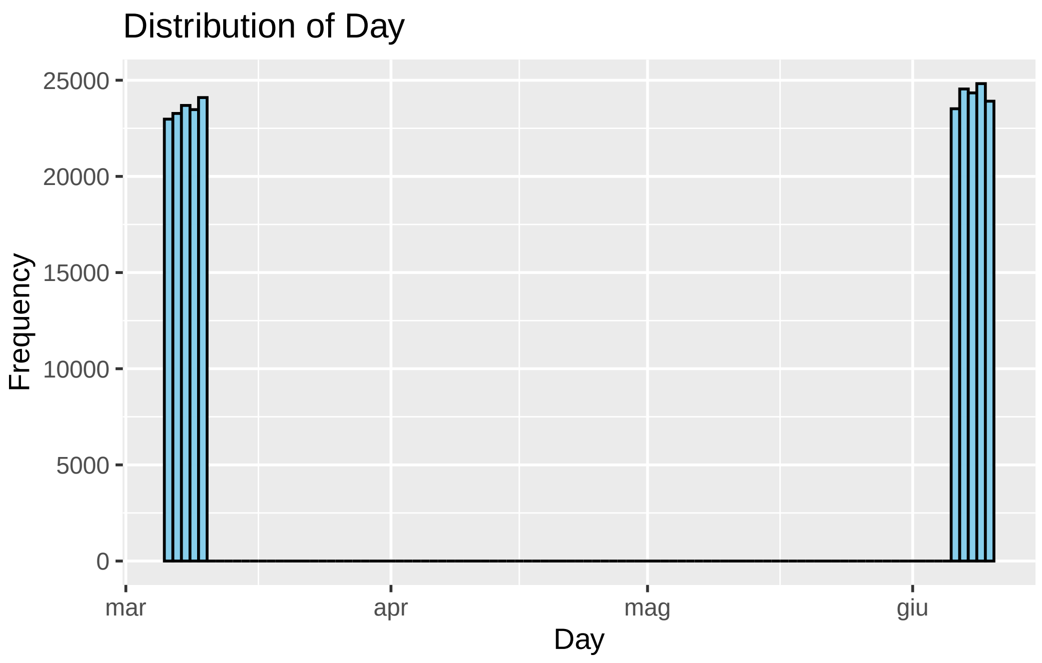
2.0.1 filter only residents

```
data <- data %>%
    filter(res_trav == "resident")
glimpse(data, width = 60)
```

```
Rows: 238,648
Columns: 7
$ day         <date> 2023-03-06, 2023-03-06, 2023-03-06, 2~
$ origin      <chr> "Porcia", "Pordenone", "Pordenone", "C~
$ destination <chr> "Pordenone", "Porcia", "Cordenons", "P~
$ n           <dbl> 1876, 1794, 1695, 1667, 1425, 1366, 13~
$ weekday     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ direction   <chr> "outbound", "inbound", "inbound", "out~
$ res_trav    <chr> "resident", "resident", "resident", "r~
```

2.0.2 check time frame

```
data %>% ggplot(aes(x = day)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Day", x = "Day", y = "Frequency")
```

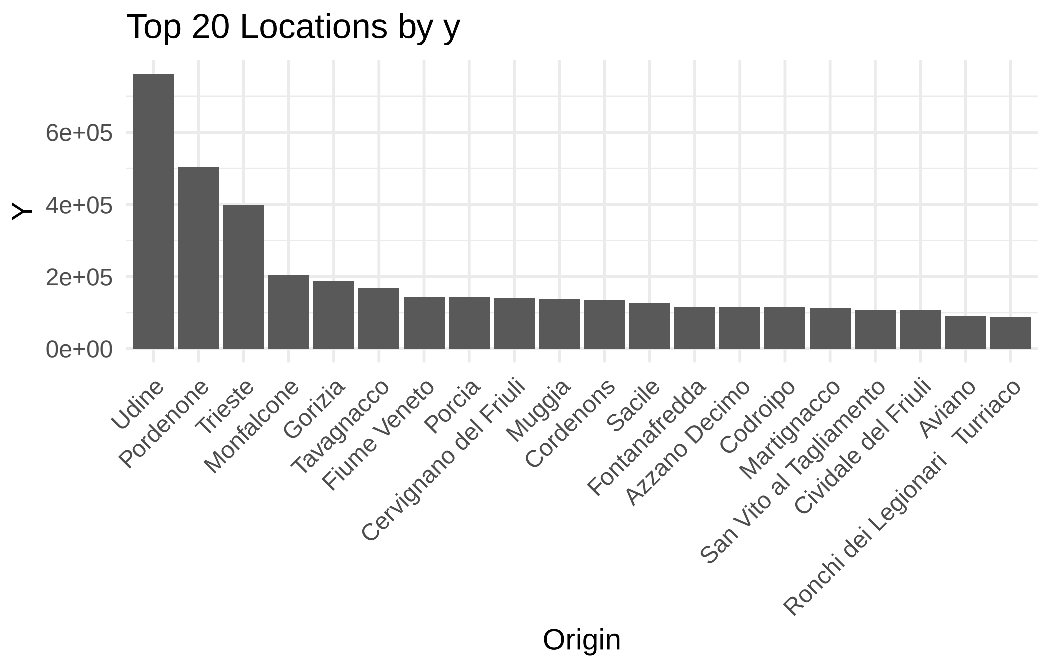


2.1 check locations

```
max_n <- max(data$n)

locations <- data %>%
  group_by(origin) %>%
  summarize(y = sum(n) ) %>%
  arrange(-y)
```

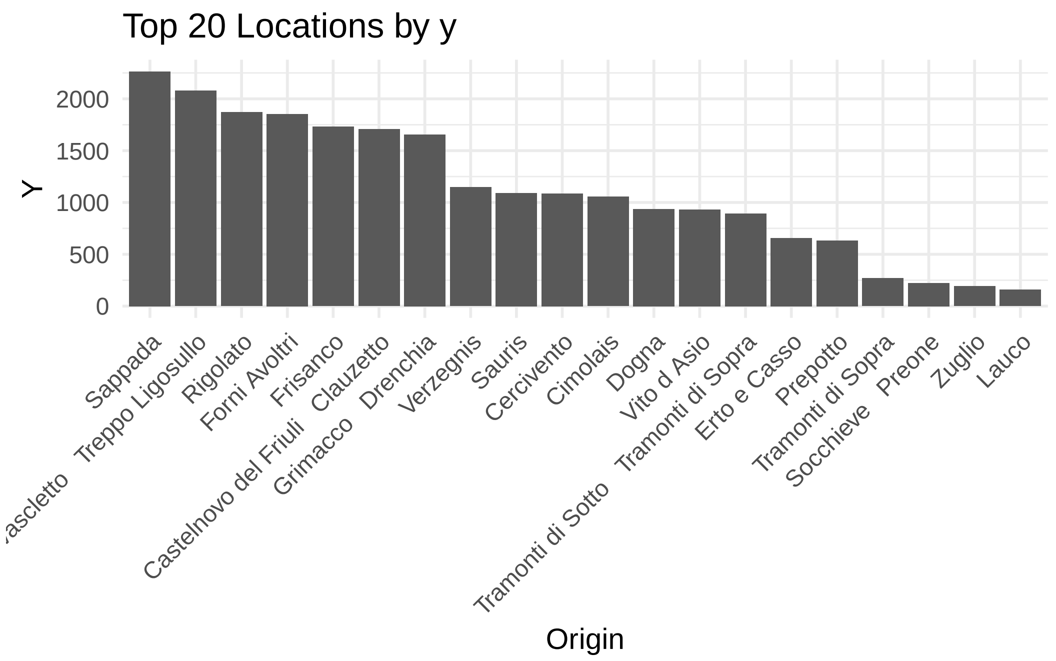
top 20

```
# Selecting top 20 locations
top_locations <- head(locations, 20)

# Creating the horizontal column plot
ggplot(top_locations, aes(x = reorder(origin, -y), y = y)) +
  geom_col() +
  labs(title = "Top 20 Locations by y",
       x = "Origin",
       y = "Y") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



last 20

```
locations %>%
    tail(20)%>%
    ggplot(aes(x = reorder(origin, -y), y = y)) +
  geom_col() +
  labs(title = "Top 20 Locations by y",
       x = "Origin",
       y = "Y") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



2.2 Improve information on locations

This section aims to enrich the data on the *origin* and *destination* with latitude, longitude, area, and province according to the ISTAT data on *comuni*.

We must take into account that some locations in the dataset are composed of two *comuni*, hence we need to summarize the information:

- mean Latitude and, longitude
- sum of area
- Name and province of the first enrty in the list.

```
comuni<-read_excel('./data/comuniFVG.xlsx') %>%
    arrange(location, nome)



#check duplicates
duplicate_locations <- comuni %>%
  group_by(location) %>%
    arrange(location)%>%
```

```r
  filter(n() > 1)

print(duplicate_locations)
```

```
# A tibble: 23 x 9
# Groups:   location [11]
   nome         nome_breve location codISTAT codCATASTALE   LAT   LON   km2 Prov
   <chr>        <chr>      <chr>    <chr>    <chr>        <dbl> <dbl> <dbl> <chr>
 1 Artegna      Artegna    Artegna~ 030006   A448          46.2  13.2  11.2 UD
 2 Montenars    Montenars  Artegna~ 030061   F574          46.3  13.2  20.6 UD
 3 Andreis      Andreis    Barcis ~ 093001   A283          46.2  12.6  27.0 PN
 4 Barcis       Barcis     Barcis ~ 093006   A640          46.2  12.6 103.  PN
 5 Buja         Buja       Buja   ~ 030013   B259          46.2  13.1  25.5 UD
 6 Treppo Gra~ TreppoG    Buja   ~ 030126   L382          46.2  13.2  11.3 UD
 7 Carlino      Carlino    Carlino~ 030018   B788          45.8  13.2  30.2 UD
 8 Marano Lag~ Marano     Carlino~ 030056   E910          45.8  13.2  85.8 UD
 9 Castelnovo~ Castelno   Casteln~ 093011   C217          46.2  12.9  22.5 PN
10 Clauzetto    Clauzetto  Casteln~ 093016   C791          46.2  12.9  28.3 PN
# i 13 more rows
```

Group and summarize locations

```r
locs <- comuni %>%
    group_by(location) %>%
    summarize(loc = first(nome_breve),
              LAT = round(mean(LAT),4),
              LON = round(mean(LON),4),
              km2 = round(sum(km2),4),
              prov = first(Prov),
              codISTAT = first(codISTAT),
              codCATASTALE = first(codCATASTALE))
```

```r
tmp <- locs %>% select(location, loc)
data <- data %>%
  left_join(tmp, by = c("origin" = "location")) %>%
  select(-origin) %>%
  rename(origin = loc) %>%
  left_join(tmp, by = c("destination" = "location")) %>%
  select(-destination) %>%
  rename(destination = loc)
```

```r
# check that comuni contains a row for each location
length(locations$origin) == length(locs$location)
```

```
[1] TRUE
```

```r
all( sort(locations$origin) == sort(locs$location) )
```

```
[1] TRUE
```

3 Saving data to csv

```r
data %>%  write_csv('./data/flows.csv')
locs %>% select(-location) %>%
    write_csv('./data/locations.csv')
```

4 Data exploration

```r
flows <-read_csv('./data/flows.csv')
```

```
Rows: 238648 Columns: 7
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (4): direction, res_trav, origin, destination
dbl  (2): n, weekday
date (1): day

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.
```

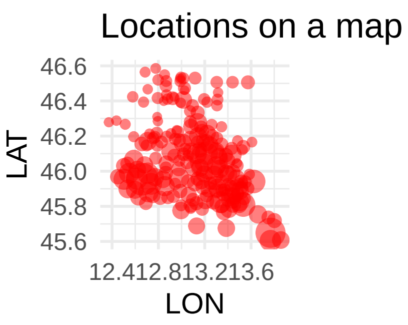
```r
locs <- read_csv('./data/locations.csv')
```

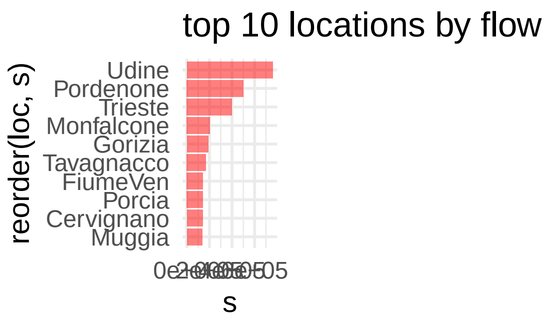
```
Rows: 203 Columns: 7
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): loc, prov, codISTAT, codCATASTALE
dbl (3): LAT, LON, km2

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.
```

```r
tmp <- flows %>%
  group_by(origin) %>%
  summarize(s = sum(n)) %>%
  rename(loc = origin)

locs <- locs %>% left_join(tmp)
```

```
Joining with `by = join_by(loc)`
```

```r
plot_points <- locs %>%
  ggplot(aes(x = LON, y = LAT)) +
  geom_point(aes(size = s), color = "red", alpha = 0.5) +
  theme_minimal() +
theme(legend.position = 'none')+
  theme(aspect.ratio = 1)+
  ggtitle("Locations on a map")

top20 <- locs

plot_head <- locs %>% arrange(-s) %>% head(10) %>%
    ggplot(aes(y = reorder(loc, s), x = s)) +
    geom_col(fill = 'red', alpha= 0.5) +
    theme_minimal() +
    theme(aspect.ratio = 2)+
    ggtitle("top 10 locations by flow")

# Arrange plots in a row

grid.arrange(plot_points, plot_head, nrow = 2)
```


Locations on a map
46.6
46.4
46.2
46.0
45.8
45.6
LAT
12.4
12.8
13.2
13.6
LON


top 10 locations by flow
Udine
Pordenone
Trieste
Monfalcone
Gorizia
Tavagnacco
FiumeVen
Porcia
Cervignano
Muggia
reorder(loc, s)
s