Lab 1

Casey Breen

2025-01-20

Lab 1: Querying the FB Marketing API

In this lab, we'll explore how to obtain FB monthly active user counts by geography, gender, and age. These data can be used to study topics such as migration or digital inequality, for instance:

- Fatehkia, Masoomali, Ridhi Kashyap, and Ingmar Weber. 2018. 'Using Facebook Ad Data to Track the Global Digital Gender Gap'. *World Development* 107:189–209. doi: 10.1016/j.worlddev.2018.03.007.
- Rampazzo, Francesco, Jakub Bijak, Agnese Vitali, Ingmar Weber, and Emilio Zagheni. 2021. 'A Framework for Estimating Migrant Stocks Using Digital Traces and Survey Data: An Application in the United Kingdom'. *Demography* 58(6):2193–2218. doi: 10.1215/00703370-9578562.

To query the Facebook Marketing API (Application Program Interface), we'll use the `rsocialwatcher` package. This R package helps us query the FB Marketing API.

The package requires your credentials and some target population (e.g., women aged 15-49 in France). The package then queries the API and returns a dataframe containing the number of Facebook monthly active users for your specified audience.

You'll need to install the `rsocialwatcher` package before working through the lab using the `install.packages("<package_name>")` command.

```
## Uncomment below line if you haven't installed rsocialwatcher or tidyverse package
# install.packages("rsocialwatcher")
# install.packages("tidyverse")

library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(rsocialwatcher)
```

Credentials

To query the Facebook Marketing API requires creating an account and obtaining credentials. This process for obtaining credentials is fairly straightforward (see instruction here) — feel free to give it a shot outside of the lab, if you're interested. Here, we'll use our existing credentials.

We'll need to load in some credentials before we get started. Specifically, we'll need the:

- API Version
- Token
- Creation account

We can load these from a separate credentials script. You'll need to update the script with the correct path to your credentials.

```r
## give path to credentials file
source("../../private/credentials.R")


#### Alternatively, directly supply credentials
# VERSION = "enter-version"
# CREATION_ACT = "creation_act"
# TOKEN = "enter-token"
```

Basic query of FB Marketing API

To query the Facebook Marketing API using the `rsocialwatcher` package, the main function we will use is: `rsocialwatcher::query_fb_marketing_api()`. The package also has other functions as well — to learn more, see the package website.

First, we can use the help to learn more about the function using the built in documentation figure.

```
?query_fb_marketing_api
```

This function clearly has lots of arguments, but let's focus on the basic functionality for now. Here's the code to query all FB users in Great Britain between the ages of 18 and 65.

```
## All Facebook users in Great Britain between ages of 18 and 65
fb_mau_users <- query_fb_marketing_api(
  location_unit_type = "countries",
  location_keys = "GB",
  age_min = 18,
  age_max = 65,
  version = VERSION,
  creation_act = CREATION_ACT,
  token = TOKEN)
```

```
No encoding supplied: defaulting to UTF-8.
```

```
fb_mau_users
```

```
  estimate_dau estimate_mau_lower_bound estimate_mau_upper_bound
1     47009608                 48100000                 56600000
  location_unit_type location_types location_keys gender age_min age_max
1          countries home or recent            GB 1 or 2      18      65
    api_call_time_utc
1 2025-01-20 21:25:27
```

Interpreting output

There key columns in the output are:

- **`estimate_dau`** = number of daily active users

- **estimate_mau_lower_bound** = lower bound estimate of monthly active users
- **estimate_mau_upper_bound** = upper bound estimate of monthly active users
- **location_keys** = country code (2-letter)
- Gender = gender (1 = male, 2 = female)
- age_min = minimum age
- age_max = maximum age

For simplicity, we'll focus on **estimate_mau_upper_bound** for the rest of this lab. This is more stable metric than daily active users, which has more day-to-day fluctuations.

Exercise 1

1. How many FB monthly active users are in France between the age of 18 and 65? (Use the **estimate_mau_upper_bound** column.)
2. How many FB monthly active users in Great Britain are between the ages of 40 and 50? Is this more or less than the number of monthly active users between 30 and 40?

```
## exercise 1.1


## exercise 1.2
```

Making multiple queries

There are several useful ways to query data for multiple countries, genders, etc. at once.

- The map_param function allows you to specify multiple countries, genders, etc. at the same time. This will return multiple rows.
- In contrast, including a vector — e.g., c("US", "MX", "CA") — returns MAU counts for all countries pooled together

```
## Query users by gender in US, MX, and CA
query_fb_marketing_api(
  location_unit_type = "countries",
  location_keys = map_param("US", "MX", "CA"),
  gender = map_param(1, 2),
  age_min = 13,
  age_max = 65,
  version = VERSION,
```

```
  creation_act = CREATION_ACT,
  token = TOKEN)
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

```
  estimate_dau estimate_mau_lower_bound estimate_mau_upper_bound
1     99163441                112500000                132400000
2     40044557                 46200000                 54400000
3     12218318                 13400000                 15800000
4    123270235                128300000                151000000
5     45603169                 49800000                 58600000
6     14587029                 15000000                 17600000
  location_unit_type location_types location_keys gender age_min age_max
1          countries home or recent            US      1      13      65
2          countries home or recent            MX      1      13      65
3          countries home or recent            CA      1      13      65
4          countries home or recent            US      2      13      65
5          countries home or recent            MX      2      13      65
6          countries home or recent            CA      2      13      65
    api_call_time_utc
1 2025-01-20 21:25:27
2 2025-01-20 21:25:27
3 2025-01-20 21:25:28
4 2025-01-20 21:25:28
5 2025-01-20 21:25:28
6 2025-01-20 21:25:29
```

Investigating age patterns

Next, let's investigate age patterns. We'll have to do this separately for each age group (`map_param` doesn't quite work here.)

Rather than writing out the same query (with different ages) lots of times, we'll use a for loop. Spend a minute trying to follow the logic here before running the code!

```r
## Define the age groups of interest
age_groups <- list(
  c(15, 19),
  c(20, 24),
  c(25, 29),
  c(30, 34),
  c(35, 39),
  c(40, 44),
  c(45, 49)
)

## Create an empty list to store results
results <- list()

# Loop through each age group and query the API
for (i in seq_along(age_groups)) {

  ##
  age_min <- age_groups[[i]][1]
  age_max <- age_groups[[i]][2]

  # Query the API for the current age group
  results[[i]] <- query_fb_marketing_api(
    location_unit_type = "countries",
    location_keys = map_param("IN"),
    gender = map_param(1, 2), # Both genders
    age_min = age_min,
    age_max = age_max,
    version = VERSION,
    creation_act = CREATION_ACT,
    token = TOKEN
  )
}
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

```r
# Combine all results into a single dataframe (if needed)
india_age_mau <- bind_rows(results)
```

Visualization monthly active user age patterns

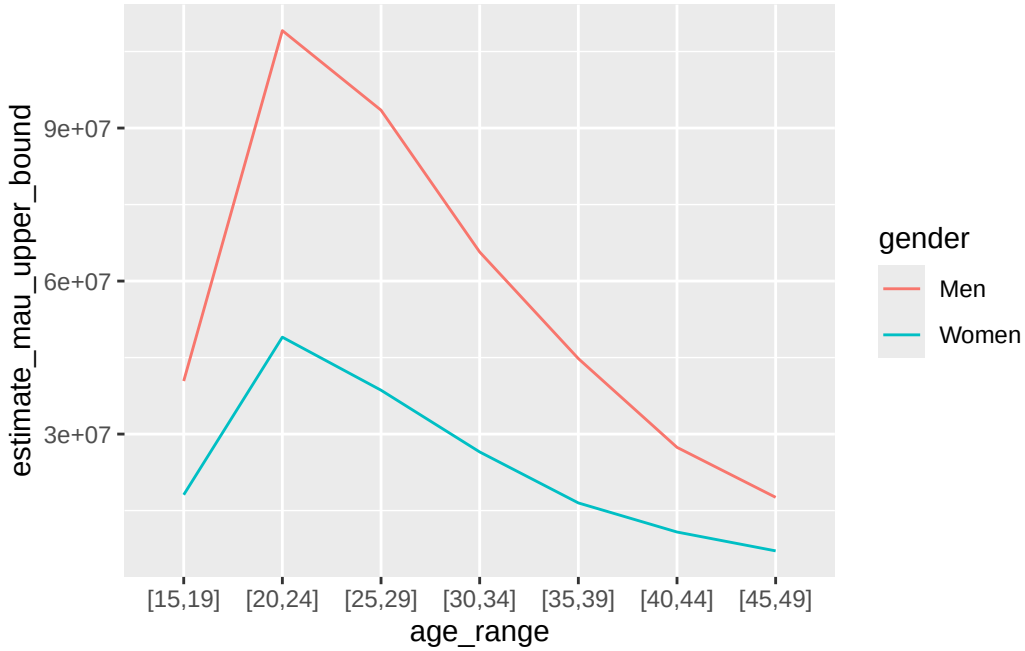
Now let's visualize age patterns in FB MAU user in India. Before running the code below, make a hypothesis about the age groups you anticipate will have the most monthly active users.

First, we'll need to create a variable corresponding to the age category. We'll also convert the gender variable from numeric to character (1 = "men", 2 = "women").

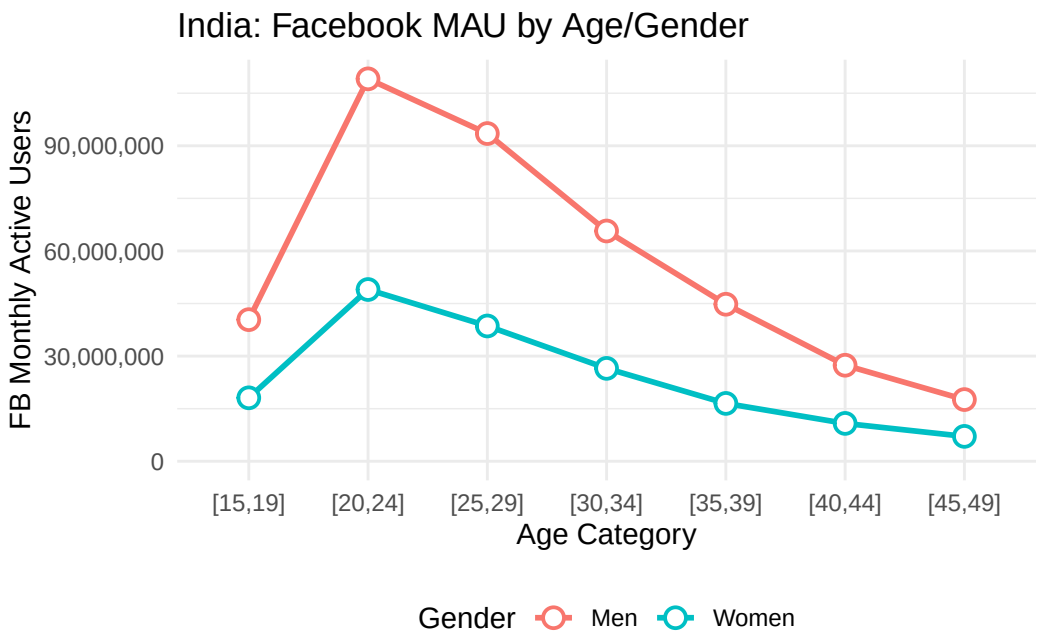
```r
# Create age categories variable from age_min and age_max variables
india_age_mau <- india_age_mau %>%
  mutate(age_range = paste0("[", age_min, ",", age_max, "]")) %>%
  mutate(gender = case_when(
    gender == 1 ~ "Men",
    gender == 2 ~ "Women"
  ))
```

We'll create two data visualizations to gain insight into the relationship between age/gender and FB usage. The first visualization is a basic plot you might make for yourself when you're doing some quick exploratory data analysis. The second is a more polished figure that you might include in a paper.

```r
## Basic plot
india_age_mau %>%
  ggplot(aes(x = age_range, y = estimate_mau_upper_bound, color = gender, group = gender)) +
  geom_line()
```




```
## Fancy plot
india_age_mau %>%
  ggplot(aes(x = age_range, y = estimate_mau_upper_bound, color = gender, group = gender)) +
  geom_line(linewidth = 1) +  # Thicker lines for clarity
  geom_point(size = 3, shape = 21, fill = "white", stroke = 1) +  # Hollow points with white
  scale_y_continuous(labels = scales::comma, limits = c(0, max(india_age_mau$estimate_mau_up
  theme_minimal() +
  labs(
    x = "Age Category",
    y = "FB Monthly Active Users",
    color = "Gender",
    title = "India: Facebook MAU by Age/Gender") +
  theme(legend.position = "bottom")
```



Exercise 2

1. Are there more women or men FB monthly active users in India? What are some potential reasons for this?
2. What age group has the fewest monthly active users? Did this align with your hypothesis?

Digital gender gaps

To what extent can Facebook tell us about gender inequality in access to the internet? In this section, we'll first query the FB marketing API to produce gender-specific MAU counts for 11 different countries. We'll then calculate basic FB gender gaps and benchmark this against internet digital gender gaps. Here, we'll define a FB gender gap as:

$$\text{FB}_{\text{Gender gap}} = \frac{\text{MAU}_{women}}{\text{MAU}_{men}}$$

First, we'll load data on internet gender gaps. These data comes from the Digital Gender Gaps projects.

```r
## internet gender gaps data
internet_data_gaps <- data.frame(
              country = c("Afghanistan","Brazil",
                          "Democratic Republic of the Congo","France","India",
                          "Japan","Nigeria","Saudi Arabia","Sweden",
                          "United States","South Africa"),
                 date = c("2024-11-01",
                          "2024-11-01","2024-11-01","2024-11-01","2024-11-01",
                          "2024-11-01","2024-11-01","2024-11-01","2024-11-01",
                          "2024-11-01","2024-11-01"),
            internet_gender_gap = c(0.472,1,0.522,0.99,
                          0.77,0.955,0.617,0.995,1,1,0.96),
                 iso2 = c("AF","BR","CD","FR",
                          "IN","JP","NG","SA","SE","US","ZA"))
```

Next, we'll query data for each country separately by gender

```r
## Query FB marketing api
all_countries_gender <- query_fb_marketing_api(
  location_unit_type = "countries",
  location_keys = map_param_vec(internet_data_gaps$iso2),
  gender = map_param(1, 2),
  version = VERSION,
  creation_act = CREATION_ACT,
  token = TOKEN)
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

```
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
No encoding supplied: defaulting to UTF-8.
```

Now we need to manipulate our data so it's in the right form for our analysis. Recoding, reshaping, and merging data is often one of the most important and time consuming steps in any data analysis project.

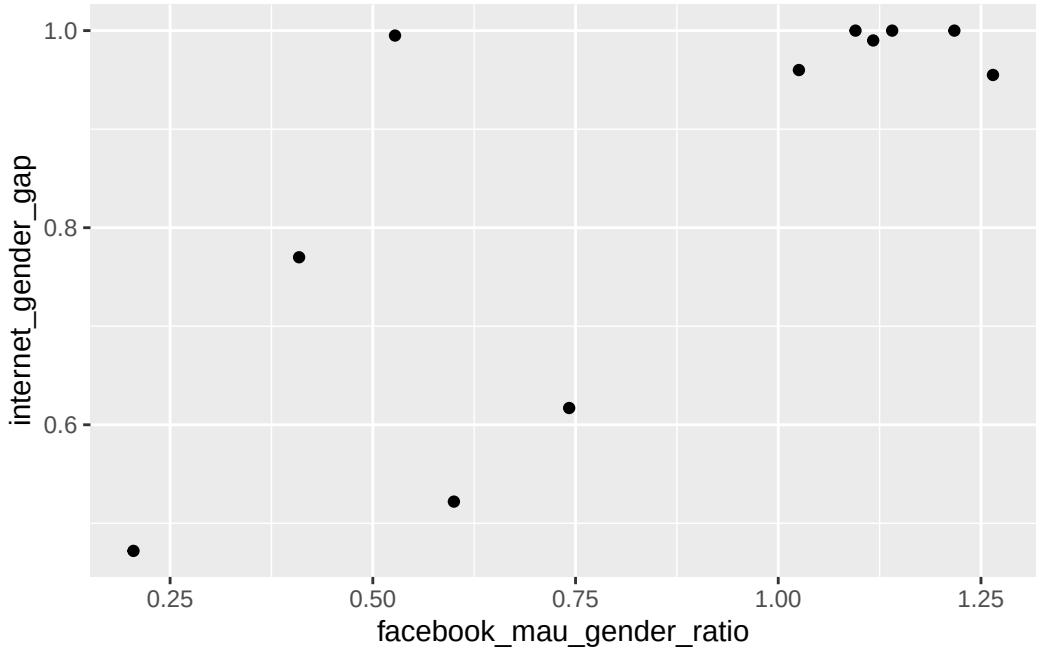
Take a minute to try to understand what each line is doing before running it.

```r
## recode gender
all_countries_gender <- all_countries_gender %>%
  mutate(gender_recode = case_when(
    gender == 1 ~ "male",
    gender == 2 ~ "female"
  ))

## reshape data from long to wide
all_countries_gender_wide <- all_countries_gender %>%
  select(location_keys, gender_recode, estimate_mau_upper_bound) %>%
  pivot_wider(names_from = gender_recode, values_from = estimate_mau_upper_bound)

## calculate gender ratios
all_countries_gender_wide <- all_countries_gender_wide %>%
  mutate(facebook_mau_gender_ratio = female/male) %>%
  left_join(internet_data_gaps, by = c("location_keys" = "iso2"))
```

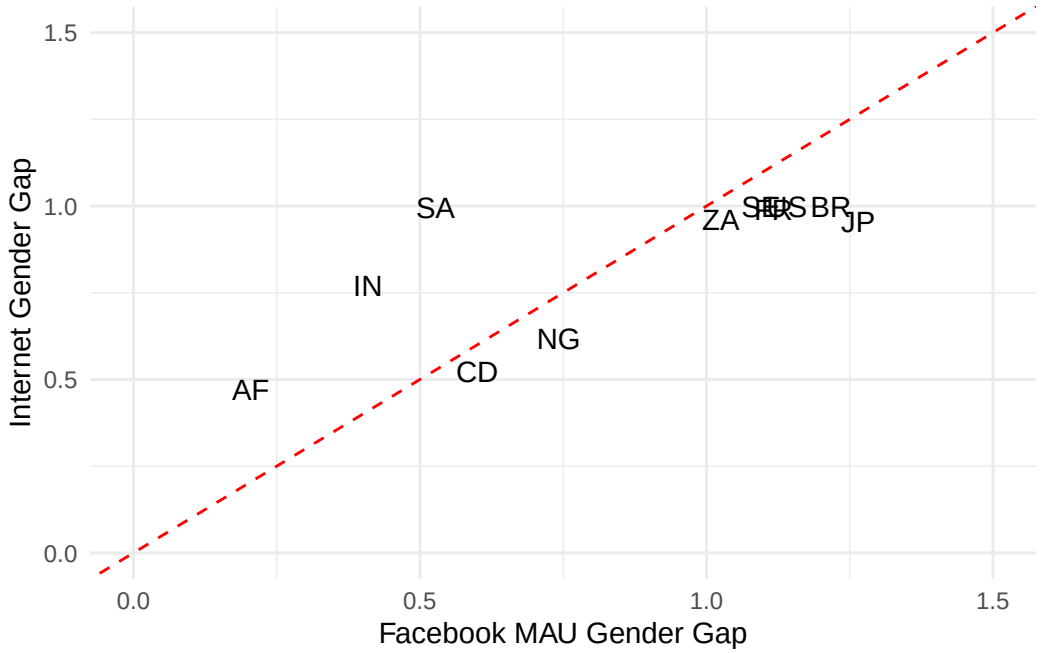
Visualize results

Now let's investigate the relationship between the FB gender gap and the overall internet gender gap. Do you think they'll be highly correlated?

```r
## Basic plot
all_countries_gender_wide %>%
  ggplot(aes(x = facebook_mau_gender_ratio, y = internet_gender_gap)) +
  geom_point()
```




```
## Fancy plot
all_countries_gender_wide %>%
  ggplot(aes(x = facebook_mau_gender_ratio, y = internet_gender_gap, label = location_keys))
  geom_text() +
  ylim(0, 1.5) +
  xlim(0, 1.5) +
  geom_abline(slope = 1, linetype = "dashed", color = "red") +
  theme_minimal() +
  labs(x = "Facebook MAU Gender Gap",
       y = "Internet Gender Gap")
```



Exercise 3

1. Calculate the correlation between the MAU gender gap and the Internet gender gap.
2. In what settings would the FB gender gap not be a good predictor of internet gender gaps? What other predictors might be used in conjunction with the Facebook gender gap data to predict internet gender gaps?

```
## 3.1
# Hint:
# all_countries_gender_wide %>%
#   summarize(cor(___, ___))
```