Lab_01_GroupG

Chathurangi Godahewa Gamage, Dury Kim , Renata King , Shantikrishna Panicker

2024-09-25

Load Necessary Libraries

```r
# Load the libraries
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
```

```r
library(readr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(arrow)  # To read Parquet files
```

```
## Warning: package 'arrow' was built under R version 4.3.3

##
## Attaching package: 'arrow'

## The following object is masked from 'package:lubridate':
##
##     duration

## The following object is masked from 'package:utils':
##
##     timestamp
```

```r
library(tidyr)
```

1. Load the data sets

```r
# Load Spark data
sp_data <- read_csv("sp_data.csv")
```

```
## Rows: 811776 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): sa2, cnt
## dttm (1): ts
##
## 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
# Load Vodafone data (parquet format)
vf_data <- read_parquet("vf_data.parquet")

# Load SA2 concordance data
sa2_data <- read_csv("sa2_2023.csv", skip = 6)
```

```
## New names:
## Rows: 2395 Columns: 3
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Descriptor dbl (1): Code lgl (1): ...3
## 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.
## * `` -> `...3`
```

```r
# Load population estimates
pop_estimates <- read_csv("subnational_pop_ests.csv")
```

```
## Rows: 4970 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): STRUCTURE, STRUCTURE_ID, STRUCTURE_NAME, ACTION, SEX_POPES_SUB_006,...
## dbl (3): YEAR_POPES_SUB_006, Year at 30 June, OBS_VALUE
## lgl (2): Area, Observation value
##
## 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
# Load the new concordance and indicator datasets
urban_rural_indicator <- read_csv("urban_rural_to_indicator_2023.csv", skip = 6)
```

```
## New names:
## Rows: 745 Columns: 4
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (3): ...2, ...3, IUR2018 V1.0.0 dbl (1): UR2023 V1.0.0
## 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.
## * `` -> `...2`
## * `` -> `...3`
```

```r
urban_rural_sa2_concord <- read_csv("urban_rural_to_sa2_concord_2023.csv", skip = 6)
```

```
## New names:
## Rows: 2815 Columns: 4
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (3): ...2, ...3, UR2023 V1.0.0 dbl (1): SA22023 V1.0.0
## 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.
## * `` -> `...2`
## * `` -> `...3`
```

```r
# Split the column under 'UR2018 V1.0.0' into two separate columns
urban_rural_indicator_cleaned <- urban_rural_indicator %>%
  separate(`IUR2018 V1.0.0`, into = c("UR2018_code", "Settlement_Type"), sep = ",") %>%
  rename(UR2023_code = `UR2023 V1.0.0`, Region_Name = `...2`, Mapping_Type = `...3`)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 745 rows [1, 2, 3, 4, 5, 6,
## 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```r
# Check the cleaned dataset
head(urban_rural_indicator_cleaned)
```

```
## # A tibble: 6 x 5
##   UR2023_code Region_Name   Mapping_Type    UR2018_code Settlement_Type
##         <dbl> <chr>         <chr>           <chr>       <chr>
## 1        1001 Pukenui       Many To One Map 21          Rural settlement
## 2        1002 Kaimaumau     Many To One Map 21          Rural settlement
## 3        1003 Tokerau Beach Many To One Map 21          Rural settlement
## 4        1004 Karikari      Many To One Map 21          Rural settlement
## 5        1005 Awanui        Many To One Map 21          Rural settlement
## 6        1006 Ahipara       Many To One Map 14          Small urban area
```

```r
# Split the column under 'UR2023 V1.0.0' into two separate columns
urban_rural_sa2_concord_cleaned <- urban_rural_sa2_concord %>%
  separate(`UR2023 V1.0.0`, into = c("UR2023_code", "Region"), sep = ",") %>%
  rename(SA2_code = `SA22023 V1.0.0`, Region_Description = `...2`, Mapping_Info = `...3`)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 2815 rows [1, 2, 3, 4, 5, 6,
## 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```r
# Check the cleaned dataset
head(urban_rural_sa2_concord_cleaned)
```

```
## # A tibble: 6 x 5
##   SA2_code Region_Description        Mapping_Info     UR2023_code Region
##      <dbl> <chr>                     <chr>            <chr>       <chr>
## 1   100100 North Cape                Many To Many Map 1001        Pukenui
## 2   100100 North Cape                Many To Many Map 1013        Other rural F~
## 3   100200 Rangaunu Harbour          Many To Many Map 1005        Awanui
## 4   100200 Rangaunu Harbour          Many To Many Map 1013        Other rural F~
## 5   100200 Rangaunu Harbour          Many To Many Map 1002        Kaimaumau
## 6   100301 Inlets Far North District Simple Map       1015        Inlets Far No~
```

```r
# Convert UR2023_code to character in urban_rural_sa2_concord_cleaned
urban_rural_sa2_concord_cleaned <- urban_rural_sa2_concord_cleaned %>%
  mutate(UR2023_code = as.character(UR2023_code))

# Convert UR2023_code to character in urban_rural_indicator_cleaned
urban_rural_indicator_cleaned <- urban_rural_indicator_cleaned %>%
  mutate(UR2023_code = as.character(UR2023_code))
```

```r
# Merge the datasets on UR2023_code
final_combined_data_with_regions <- urban_rural_sa2_concord_cleaned %>%
  left_join(urban_rural_indicator_cleaned, by = "UR2023_code")
```

```r
# Check the merged result
head(final_combined_data_with_regions)
```

```
## # A tibble: 6 x 9
##   SA2_code Region_Description        Mapping_Info UR2023_code Region Region_Name
##      <dbl> <chr>                     <chr>        <chr>       <chr>  <chr>
## 1   100100 North Cape                Many To Man~ 1001        Puken~ Pukenui
## 2   100100 North Cape                Many To Man~ 1013        Other~ Other rura~
## 3   100200 Rangaunu Harbour          Many To Man~ 1005        Awanui Awanui
## 4   100200 Rangaunu Harbour          Many To Man~ 1013        Other~ Other rura~
## 5   100200 Rangaunu Harbour          Many To Man~ 1002        Kaima~ Kaimaumau
## 6   100301 Inlets Far North District Simple Map   1015        Inlet~ Inlets Far~
## # i 3 more variables: Mapping_Type <chr>, UR2018_code <chr>,
## #   Settlement_Type <chr>
```

2. Clean and Process Data Process Timestamps and Cleaning Missing Data

```r
# Clean Spark Data
sp_data <- sp_data %>%
  mutate(ts = as_datetime(ts),  # Convert timestamp to datetime
         cnt = replace_na(cnt, 0))  # Replace NA values in count
head(sp_data)
```

```
## # A tibble: 6 x 3
##   ts                     sa2   cnt
##   <dttm>               <dbl> <dbl>
## 1 2024-06-02 12:00:00 100100  793.
## 2 2024-06-02 13:00:00 100100  742.
## 3 2024-06-02 14:00:00 100100 1233.
## 4 2024-06-02 15:00:00 100100  959.
## 5 2024-06-02 16:00:00 100100 1134.
## 6 2024-06-02 17:00:00 100100  663.
```

```r
# Clean and process Vodafone data
vf_data <- vf_data %>%
  mutate(dt = as_datetime(dt),  # Convert dt to proper datetime format
         devices = replace_na(devices, 0))  # Replace NA values in devices column with 0

# Check the cleaned data
head(vf_data)
```

```
## # A tibble: 6 x 3
##   dt                  area   devices
##   <dttm>              <chr>    <dbl>
## 1 2024-06-03 00:00:00 100100    340.
## 2 2024-06-03 01:00:00 100100    318.
## 3 2024-06-03 02:00:00 100100    528.
## 4 2024-06-03 03:00:00 100100    411.
## 5 2024-06-03 04:00:00 100100    486.
## 6 2024-06-03 05:00:00 100100    284.
```

Merge with SA2 Codes sp_data set

```r
# Remove the '...3' column by selecting only the relevant columns
sa2_data <- sa2_data %>%
  select(Code, Descriptor)
```

```r
# Check the result
colnames(sa2_data)
```

```
## [1] "Code"       "Descriptor"
```

```r
head(sa2_data)
```

```
## # A tibble: 6 x 2
##      Code Descriptor
##     <dbl> <chr>
## 1 100100 North Cape
## 2 100200 Rangaunu Harbour
## 3 100301 Inlets Far North District
## 4 100400 Karikari Peninsula
## 5 100500 Tangonge
## 6 100600 Ahipara
```

First Ensure merging data types

```r
# Check the data type of sp_data$sa2
str(sp_data$sa2)
```

```
##  num [1:811776] 1e+05 1e+05 1e+05 1e+05 1e+05 ...
```

```r
# If sp_data$sa2 is not numeric, convert it to numeric
sp_data <- sp_data %>%
  mutate(sa2 = as.numeric(sa2))
```

```r
# Merge sp_data with sa2_data using the SA2 codes
sp_data_merged <- sp_data %>%
  left_join(sa2_data, by = c("sa2" = "Code"))

# Check the merged result
head(sp_data_merged)
```

```
## # A tibble: 6 x 4
##   ts                     sa2   cnt Descriptor
##   <dttm>               <dbl> <dbl> <chr>
## 1 2024-06-02 12:00:00 100100  793. North Cape
## 2 2024-06-02 13:00:00 100100  742. North Cape
## 3 2024-06-02 14:00:00 100100 1233. North Cape
## 4 2024-06-02 15:00:00 100100  959. North Cape
## 5 2024-06-02 16:00:00 100100 1134. North Cape
## 6 2024-06-02 17:00:00 100100  663. North Cape
```

Merge with SA2 codes with vf_dataset

```r
# Check the column names in vf_data
colnames(vf_data)
```

```
## [1] "dt"      "area"    "devices"
```

```r
str(vf_data)
```

```
## tibble [811,752 x 3] (S3: tbl_df/tbl/data.frame)
##  $ dt     : POSIXct[1:811752], format: "2024-06-03 00:00:00" "2024-06-03 01:00:00" ...
##  $ area   : chr [1:811752] "100100" "100100" "100100" "100100" ...
##  $ devices: num [1:811752] 340 318 528 411 486 ...
```

```r
# Convert area in vf_data to numeric (if it isn't already)
vf_data <- vf_data %>%
  mutate(area = as.numeric(area))
```

Perform merge

```r
# Merge vf_data with sa2_data using the SA2 codes
vf_data_merged <- vf_data %>%
  left_join(sa2_data, by = c("area" = "Code"))

# Check the merged result
head(vf_data_merged)
```

```
## # A tibble: 6 x 4
##   dt                    area devices Descriptor
##   <dttm>               <dbl>   <dbl> <chr>
## 1 2024-06-03 00:00:00 100100    340. North Cape
## 2 2024-06-03 01:00:00 100100    318. North Cape
## 3 2024-06-03 02:00:00 100100    528. North Cape
## 4 2024-06-03 03:00:00 100100    411. North Cape
## 5 2024-06-03 04:00:00 100100    486. North Cape
## 6 2024-06-03 05:00:00 100100    284. North Cape
```

3. Aggregate Data for population count

Aggregate the population counts based on the SA2 code and datetime.

```r
# Aggregate Spark data by SA2 and timestamp, dropping groups after summarizing
sp_data_aggregated <- sp_data_merged %>%
  group_by(sa2, ts) %>%
  summarise(count = sum(cnt, na.rm = TRUE), .groups = "drop")

# Aggregate Vodafone data similarly, dropping groups after summarizing
vf_data_aggregated <- vf_data_merged %>%
  group_by(area, dt) %>%
  summarise(count = sum(devices, na.rm = TRUE), .groups = "drop")
```

Combine Spark and Vodafone Data

```r
# Merge the Spark and Vodafone data on SA2 code and timestamp
combined_data <- full_join(
  sp_data_aggregated, vf_data_aggregated,
  by = c("sa2" = "area", "ts" = "dt")
)

# Check the result
head(combined_data)
```

```
## # A tibble: 6 x 4
##      sa2 ts                  count.x count.y
##    <dbl> <dttm>                <dbl>   <dbl>
## 1 100100 2024-06-02 12:00:00    793.    340.
## 2 100100 2024-06-02 13:00:00    742.    318.
## 3 100100 2024-06-02 14:00:00   1233.    528.
## 4 100100 2024-06-02 15:00:00    959.    411.
## 5 100100 2024-06-02 16:00:00   1134.    486.
## 6 100100 2024-06-02 17:00:00    663.    284.
```

Handling missing value after combining

```r
# Replace NAs with 0 in the combined count columns
combined_data <- combined_data %>%
  mutate(count.x = replace_na(count.x, 0),  # Spark count
         count.y = replace_na(count.y, 0))  # Vodafone count

# Optionally,  can create a total count column (sum of Spark and Vodafone counts)
combined_data <- combined_data %>%
  mutate(total_count = count.x + count.y)

# Check the result
head(combined_data)
```

```
## # A tibble: 6 x 5
##      sa2 ts                  count.x count.y total_count
##    <dbl> <dttm>                <dbl>   <dbl>       <dbl>
## 1 100100 2024-06-02 12:00:00    793.    340.       1133.
## 2 100100 2024-06-02 13:00:00    742.    318.       1059.
## 3 100100 2024-06-02 14:00:00   1233.    528.       1761.
## 4 100100 2024-06-02 15:00:00    959.    411.       1371.
## 5 100100 2024-06-02 16:00:00   1134.    486.       1620.
## 6 100100 2024-06-02 17:00:00    663.    284.        947.
```

Before merging, ensure that the sa2 column in combined_data and the SA2_code column in urban_rural_sa2_concord_cleaned have the same data type

```r
# Convert sa2 to character in combined_data
combined_data <- combined_data %>%
  mutate(sa2 = as.character(sa2))

# Convert SA2_code to character in urban_rural_sa2_concord_cleaned
urban_rural_sa2_concord_cleaned <- urban_rural_sa2_concord_cleaned %>%
  mutate(SA2_code = as.character(SA2_code))
```

Merge combined_data with urban_rural_sa2_concord_cleaned

```r
# Merge final_combined_data with urban_rural_sa2_concord_cleaned
final_combined_with_concord <- combined_data %>%
  left_join(urban_rural_sa2_concord_cleaned, by = c("sa2" = "SA2_code"))
```

```
## Warning in left_join(., urban_rural_sa2_concord_cleaned, by = c(sa2 = "SA2_code")): Detected an unex
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
# Check the merged result
head(final_combined_with_concord)
```

```
## # A tibble: 6 x 9
##   sa2    ts                  count.x count.y total_count Region_Description
##   <chr>  <dttm>                <dbl>   <dbl>       <dbl> <chr>
## 1 100100 2024-06-02 12:00:00    793.    340.       1133. North Cape
## 2 100100 2024-06-02 12:00:00    793.    340.       1133. North Cape
## 3 100100 2024-06-02 13:00:00    742.    318.       1059. North Cape
## 4 100100 2024-06-02 13:00:00    742.    318.       1059. North Cape
## 5 100100 2024-06-02 14:00:00   1233.    528.       1761. North Cape
## 6 100100 2024-06-02 14:00:00   1233.    528.       1761. North Cape
## # i 3 more variables: Mapping_Info <chr>, UR2023_code <chr>, Region <chr>
```

```
# Merge the combined dataset with urban_rural_indicator_cleaned
final_combined_with_regions <- final_combined_with_concord %>%
  left_join(urban_rural_indicator_cleaned, by = "UR2023_code")

# Check the final result
head(final_combined_with_regions)
```

```
## # A tibble: 6 x 13
##   sa2    ts                  count.x count.y total_count Region_Description
##   <chr>  <dttm>                <dbl>   <dbl>       <dbl> <chr>
## 1 100100 2024-06-02 12:00:00    793.    340.       1133. North Cape
## 2 100100 2024-06-02 12:00:00    793.    340.       1133. North Cape
## 3 100100 2024-06-02 13:00:00    742.    318.       1059. North Cape
## 4 100100 2024-06-02 13:00:00    742.    318.       1059. North Cape
## 5 100100 2024-06-02 14:00:00   1233.    528.       1761. North Cape
## 6 100100 2024-06-02 14:00:00   1233.    528.       1761. North Cape
## # i 7 more variables: Mapping_Info <chr>, UR2023_code <chr>, Region <chr>,
## #   Region_Name <chr>, Mapping_Type <chr>, UR2018_code <chr>,
## #   Settlement_Type <chr>
```

```
# Keep only the necessary columns for the final dataset
final_cleaned_data <- final_combined_with_regions %>%
  select(sa2, ts,count.x, count.y, total_count, UR2023_code, UR2018_code) %>%  # Select relevant column
  distinct()  # Remove any duplicate rows

# Check the cleaned result
head(final_cleaned_data)
```

```
## # A tibble: 6 x 7
##   sa2    ts                  count.x count.y total_count UR2023_code UR2018_code
##   <chr>  <dttm>                <dbl>   <dbl>       <dbl> <chr>       <chr>
## 1 100100 2024-06-02 12:00:00    793.    340.       1133. 1001        21
## 2 100100 2024-06-02 12:00:00    793.    340.       1133. 1013        22
## 3 100100 2024-06-02 13:00:00    742.    318.       1059. 1001        21
## 4 100100 2024-06-02 13:00:00    742.    318.       1059. 1013        22
## 5 100100 2024-06-02 14:00:00   1233.    528.       1761. 1001        21
## 6 100100 2024-06-02 14:00:00   1233.    528.       1761. 1013        22
```

Exporting data as gzip

```
# Export the final dataset with regions as a gzipped CSV
write_csv(final_cleaned_data, gzfile("final_cleaned_data.csv.gz"))
```