Assignment II

Arun Prakash Krishnasamy

2025-06-04

Contents

Objectives

Electorates won by Independent candidates in 2022 federal election

The data required to answer the task is available in the **Australian Electoral Commission AEC**.

All data about the past election is available in the **Information centre**.

- The pathway for navigating the results data is as below:

  Australian Electoral Commission > Information centre > Federal election > 2022 Federal election > 2022 Federal election results

Different sets of data are accessible in the "Tally Room" download section, covering information on Candidates & seats, First preferences, two candidate preferred, two party preferred, Distribution, flow of preferences, Enrollment, informal votes, turnout, and total votes.

To find the number of electorates won by Independent party in 2022, we use the **Distribution of preferences by candidate by division**, because this provides us with the information on which party got elected in a particular division.

```
# Download link for the data
url = "https://results.aec.gov.au/27966/Website/Downloads/HouseDopByDivisionDownload-27966.csv"

fed_election_2022 <- read_csv(url, skip = 1) # Skip the first row as it contains metadata information
```

```
# Select winning Independent candidates
electorates_IND <- fed_election_2022 |>
  filter(PartyNm == "Independent" &
           Elected == "Y" &
           CalculationType == "Preference Count" &
           CountNumber == 0) |>
  select(PartyAb, DivisionNm) |>
  distinct(DivisionNm)
electorates_IND
```

```
## # A tibble: 10 x 1
##    DivisionNm
##    <chr>
##  1 Fowler
##  2 Mackellar
##  3 North Sydney
##  4 Warringah
##  5 Wentworth
##  6 Clark
##  7 Goldstein
##  8 Indi
##  9 Kooyong
## 10 Curtin
```

Total of ten electorates seats were won by the Independent candidates, namely Fowler, Mackellar, North Sydney, Warringah, Wentworth, Clark, Goldstein, Indi, Kooyong and Curtin.

Formerly Liberal seats

For this the same dataset, **Distribution of preferences by candidate by division** , can be used, as it also includes information on which party previously held each seat.

Initially, the dataset is filtered to have only the seats won by the liberals in the previous election.

```
formerly_liberal <-  fed_election_2022 |>
  filter(PartyNm == "Liberal" &
           HistoricElected == "Y" &
           Elected == "N") |>
  select(PartyAb, DivisionNm) |>
  distinct(DivisionNm)
formerly_liberal
```

```
## # A tibble: 13 x 1
##    DivisionNm
##    <chr>
##  1 Mackellar
##  2 North Sydney
##  3 Reid
##  4 Robertson
##  5 Wentworth
##  6 Chisholm
##  7 Goldstein
##  8 Higgins
##  9 Kooyong
## 10 Cowan
## 11 Curtin
## 12 Hasluck
## 13 Tangney
```

To have only the seats lost to Independent candidates `inner_join()` is used, as it only returns matching observations in both variables.

```
lost_liberal_seats <- inner_join(formerly_liberal,
                                 electorates_IND, by = "DivisionNm")
lost_liberal_seats
```

```
## # A tibble: 6 x 1
```

```
##   DivisionNm
##   <chr>
## 1 Mackellar
## 2 North Sydney
## 3 Wentworth
## 4 Goldstein
## 5 Kooyong
## 6 Curtin
```

Liberal party lost **Six Electorates** to the Independent party, namely Mackellar, North Sydney, Wentworth, Goldstein, Kooyong and Curtin.

Let us plot some distribution graphs with respect to age, gender and income in one of the electorate won by the Independent.

Choosing Goldstein as the electorate

The data on age, income and gender for Goldstein should be imported:

Data acquisition:

The information on age, gender and income of the people is available in **Australian Bureau of Statistics**.

Go to the Census section on the dashboard and click on "Find Census data." Here, you'll find numerous methods to locate the dataset you're looking for. Under "Find Census data," on the left-hand side, select "Census data tools," then choose "Data packs."

- The navigational pathway is as below:

  Australian Bureau of Statistics > Census > Find Census data > Census data tools > Data packs.

DataPacks contain comprehensive data about people, families and dwellings for all Census geographies ranging from Australia down to Statistical Area Level 1 (SA1).

They are containers of data based on Census Community Profiles and include digital boundary information, metadata and reference documents to enable you to read the data.

There is overwhelming number of data packs available, but with the help of the metadata available, containing unique ID for a particular topic, it is clear that **G17** has data on **"Total personal income (weekly) by age by sex"**.

The G17 consist of three sub divisional data for male, female and person,i.e., G17A, G17B and G17C. To read all at once, `glue()` can be used.

The `glue::glue()` is used to import all the sub divisional data, essentially, the glue::glue() function is used to insert the values of "geo" and "alpha" into the file path template and it creates paths for each value of alpha in the sequence.

```r
# setting up path (for the particular division)
census_path_CED <- here::here("data/2021_GCP_all_for_VIC_short-header/2021 Census GCP All Geographies fo

# setting up path (for the entire state Victoria)
census_path_VIC <- here::here("data/2021_GCP_all_for_VIC_short-header/2021 Census GCP States and Territo

CED_paths <- glue::glue(census_path_CED, "/{geo}/VIC/2021Census_G17{alpha}_VIC_CED.csv",
                        geo = "CED", alpha = c("A", "B", "C"))

VIC_paths <- glue::glue(census_path_VIC, "/{geo}/2021Census_G17{alpha}_VIC_STE.csv",
                        geo = "STE", alpha = c("A", "B", "C"))
```

```r
CED_G17A <- read_csv(CED_paths[1])
CED_G17B <- read_csv(CED_paths[2])
CED_G17C <- read_csv(CED_paths[3])

VIC_G17A <- read_csv(VIC_paths[1])
VIC_G17B <- read_csv(VIC_paths[2])
VIC_G17C <- read_csv(VIC_paths[3])

head(CED_G17A)
```

```
## # A tibble: 6 x 201
##   CED_CODE_2021 M_Neg_Nil_income_15_19_yrs M_Neg_Nil_income_20_24_yrs
##   <chr>                              <dbl>                      <dbl>
## 1 CED201                              2418                        526
## 2 CED202                              1823                        248
## 3 CED203                              1610                        258
## 4 CED204                              3012                        668
## 5 CED205                              2941                        718
## 6 CED206                              2163                        350
## # i 198 more variables: M_Neg_Nil_income_25_34_yrs <dbl>,
## #   M_Neg_Nil_income_35_44_yrs <dbl>, M_Neg_Nil_income_45_54_yrs <dbl>,
## #   M_Neg_Nil_income_55_64_yrs <dbl>, M_Neg_Nil_income_65_74_yrs <dbl>,
## #   M_Neg_Nil_income_75_84_yrs <dbl>, M_Negtve_Nil_incme_85_yrs_ovr <dbl>,
## #   M_Neg_Nil_income_Tot <dbl>, M_1_149_15_19_yrs <dbl>,
## #   M_1_149_20_24_yrs <dbl>, M_1_149_25_34_yrs <dbl>, M_1_149_35_44_yrs <dbl>,
## #   M_1_149_45_54_yrs <dbl>, M_1_149_55_64_yrs <dbl>, ...
```

```r
head(CED_G17B)
```

```
## # A tibble: 6 x 201
##   CED_CODE_2021 F_300_399_15_19_yrs F_300_399_20_24_yrs F_300_399_25_34_yrs
##   <chr>                       <dbl>               <dbl>               <dbl>
## 1 CED201                        253                 451                 412
## 2 CED202                        232                 437                 511
## 3 CED203                        232                 374                 487
## 4 CED204                        232                 539                 671
## 5 CED205                        279                 575                1036
## 6 CED206                        235                 402                 412
## # i 197 more variables: F_300_399_35_44_yrs <dbl>, F_300_399_45_54_yrs <dbl>,
## #   F_300_399_55_64_yrs <dbl>, F_300_399_65_74_yrs <dbl>,
## #   F_300_399_75_84_yrs <dbl>, F_300_399_85ov <dbl>, F_300_399_Tot <dbl>,
## #   F_400_499_15_19_yrs <dbl>, F_400_499_20_24_yrs <dbl>,
## #   F_400_499_25_34_yrs <dbl>, F_400_499_35_44_yrs <dbl>,
## #   F_400_499_45_54_yrs <dbl>, F_400_499_55_64_yrs <dbl>,
## #   F_400_499_65_74_yrs <dbl>, F_400_499_75_84_yrs <dbl>, ...
```

```r
head(CED_G17C)
```

```
## # A tibble: 6 x 111
##   CED_CODE_2021 P_650_799_15_19_yrs P_650_799_20_24_yrs P_650_799_25_34_yrs
##   <chr>                       <dbl>               <dbl>               <dbl>
## 1 CED201                        181                1061                1413
## 2 CED202                        265                1180                1628
## 3 CED203                        269                1079                1587
## 4 CED204                        205                1295                2110
```

```
## 5 CED205                        187                1345                2587
## 6 CED206                        243                1162                1433
## # i 107 more variables: P_650_799_35_44_yrs <dbl>, P_650_799_45_54_yrs <dbl>,
## #   P_650_799_55_64_yrs <dbl>, P_650_799_65_74_yrs <dbl>,
## #   P_650_799_75_84_yrs <dbl>, P_650_799_85ov <dbl>, P_650_799_Tot <dbl>,
## #   P_800_999_15_19_yrs <dbl>, P_800_999_20_24_yrs <dbl>,
## #   P_800_999_25_34_yrs <dbl>, P_800_999_35_44_yrs <dbl>,
## #   P_800_999_45_54_yrs <dbl>, P_800_999_55_64_yrs <dbl>,
## #   P_800_999_65_74_yrs <dbl>, P_800_999_75_84_yrs <dbl>, ...
```

```r
head(VIC_G17A)
```

```
## # A tibble: 1 x 201
##   STE_CODE_2021 M_Neg_Nil_income_15_19_yrs M_Neg_Nil_income_20_24_yrs
##           <dbl>                      <dbl>                      <dbl>
## 1             2                      88386                      21186
## # i 198 more variables: M_Neg_Nil_income_25_34_yrs <dbl>,
## #   M_Neg_Nil_income_35_44_yrs <dbl>, M_Neg_Nil_income_45_54_yrs <dbl>,
## #   M_Neg_Nil_income_55_64_yrs <dbl>, M_Neg_Nil_income_65_74_yrs <dbl>,
## #   M_Neg_Nil_income_75_84_yrs <dbl>, M_Negtve_Nil_incme_85_yrs_ovr <dbl>,
## #   M_Neg_Nil_income_Tot <dbl>, M_1_149_15_19_yrs <dbl>,
## #   M_1_149_20_24_yrs <dbl>, M_1_149_25_34_yrs <dbl>, M_1_149_35_44_yrs <dbl>,
## #   M_1_149_45_54_yrs <dbl>, M_1_149_55_64_yrs <dbl>, ...
```

```r
head(VIC_G17B)
```

```
## # A tibble: 1 x 201
##   STE_CODE_2021 F_300_399_15_19_yrs F_300_399_20_24_yrs F_300_399_25_34_yrs
##           <dbl>               <dbl>               <dbl>               <dbl>
## 1             2                8810               19537               21502
## # i 197 more variables: F_300_399_35_44_yrs <dbl>, F_300_399_45_54_yrs <dbl>,
## #   F_300_399_55_64_yrs <dbl>, F_300_399_65_74_yrs <dbl>,
## #   F_300_399_75_84_yrs <dbl>, F_300_399_85ov <dbl>, F_300_399_Tot <dbl>,
## #   F_400_499_15_19_yrs <dbl>, F_400_499_20_24_yrs <dbl>,
## #   F_400_499_25_34_yrs <dbl>, F_400_499_35_44_yrs <dbl>,
## #   F_400_499_45_54_yrs <dbl>, F_400_499_55_64_yrs <dbl>,
## #   F_400_499_65_74_yrs <dbl>, F_400_499_75_84_yrs <dbl>, ...
```

```r
head(VIC_G17C)
```

```
## # A tibble: 1 x 111
##   STE_CODE_2021 P_650_799_15_19_yrs P_650_799_20_24_yrs P_650_799_25_34_yrs
##           <dbl>               <dbl>               <dbl>               <dbl>
## 1             2                7670               45029               71776
## # i 107 more variables: P_650_799_35_44_yrs <dbl>, P_650_799_45_54_yrs <dbl>,
## #   P_650_799_55_64_yrs <dbl>, P_650_799_65_74_yrs <dbl>,
## #   P_650_799_75_84_yrs <dbl>, P_650_799_85ov <dbl>, P_650_799_Tot <dbl>,
## #   P_800_999_15_19_yrs <dbl>, P_800_999_20_24_yrs <dbl>,
## #   P_800_999_25_34_yrs <dbl>, P_800_999_35_44_yrs <dbl>,
## #   P_800_999_45_54_yrs <dbl>, P_800_999_55_64_yrs <dbl>,
## #   P_800_999_65_74_yrs <dbl>, P_800_999_75_84_yrs <dbl>, ...
```

A function can be created, so as to avoid repeating the code logic and ensure conciseness and reproducibility ,to do the following task:

- Transforming the dataset from wide to long format will enhance readability and usability by consolidating multiple variables into a single column, making data manipulation and analysis simpler.

- Combines dataset G17A, G17B, and G17C into a single dataset using bind_rows().
- Replaces inappropriate naming conventions using `string_replace`.
- Expanding the dataset to separate different values into each variable.

This function can also work for the STE dataset, which is used for comparison purposes.

```
process_dataset <- function(dataset) {
  dataset_long <- dataset |>
    pivot_longer(cols = -1, names_to = "classification", values_to = "count")

  dataset_long_processed <- dataset_long |>
    filter(!str_detect(string = classification, pattern = "Tot"),
           !str_detect(classification, pattern = "PI_NS")) |>
    mutate(
      classification = str_replace(classification, "Neg_Nil_income", "-Inf_0"),
      classification = str_replace(classification, "Neg_Nil_incme", "-Inf_0"),
      classification = str_replace(classification, "Negtve_Nil_incme", "-Inf_0"),
      classification = str_replace(classification, "more", "Inf"),
      classification = str_replace(classification, "85ov", "85_110_yrs"),
      classification = str_replace(classification, "85_yrs_ovr", "85_110_yrs"))

  dataset_tidy <- dataset_long_processed |>
    mutate(classification = str_remove(classification, "_yrs")) |>
    separate_wider_delim(cols = classification, delim = "_",
                         names = c("sex", "income_min", "income_max", "age_min", "age_max"))

  return(dataset_tidy)
}

# List of datasets
dataset_1 <- list(CED_G17A, CED_G17B, CED_G17C)
dataset_2 <- list(VIC_G17A, VIC_G17B, VIC_G17C)

# Apply the function to each dataset and combine them
CED_G17_tidy <- dataset_1 |>
  lapply(process_dataset) |>
  bind_rows()

VIC_tidy <- dataset_2 |>
  lapply(process_dataset) |>
  bind_rows()
```

The data now includes all the Commonwealth Electoral Divisions (CED), but for this analysis, Goldstein is the focus. However, the final dataset doesn't have specific information about Goldstein, only a **"CED code"** is available.

Specific CED codes for locations is available in the excel file named "2021Census_geog_desc_1st_2nd_3rd_release", in the Metadata folder.

```
# Reading the metadata
CED_code_Goldstein <- read_excel("data/2021_GCP_all_for_VIC_short-header/2021 Census GCP All Geographie

CED_code_Goldstein <- CED_code_Goldstein |>
  filter(Census_Name_2021 == "Goldstein") |>
  select(Census_Code_2021, Census_Name_2021)
```

```r
CED_code_Goldstein
```

```
## # A tibble: 1 x 2
##   Census_Code_2021 Census_Name_2021
##   <chr>            <chr>
## 1 CED217           Goldstein
```

Furthermore, you can easily find the area code for Goldstein by searching on the ABS website. As a result, the area code for Goldstein is **CED217**.

Similarly to get the code for Victoria,

```r
STE_code_VIC <- read_excel("data/2021_GCP_all_for_VIC_short-header/2021 Census GCP All Geographies for V
                                  sheet = "2021_ASGS_MAIN_Structures") # sheet that contains STE codes
STE_code_VIC <- STE_code_VIC |>
   filter(Census_Name_2021 == "Victoria") |>
  select(Census_Code_2021, Census_Name_2021)
STE_code_VIC
```

```
## # A tibble: 1 x 2
##   Census_Code_2021 Census_Name_2021
##   <chr>            <chr>
## 1 2                Victoria
```

A)Gender Distribution:

```r
CED_G17_tidy <- CED_G17_tidy |>
  unite("income", c(income_min, income_max), remove = FALSE) |>
  unite("age", c(age_min, age_max), remove = FALSE) |>
  filter(CED_CODE_2021 == "CED217") # negating electoral divisions other than Goldstein

VIC_tidy <- VIC_tidy |>
  unite("income", c(income_min, income_max), remove = FALSE) |>
  unite("age", c(age_min, age_max), remove = FALSE)

head(CED_G17_tidy)
```

```
## # A tibble: 6 x 9
##   CED_CODE_2021 sex   income income_min income_max age   age_min age_max count
##   <chr>         <chr> <chr>  <chr>      <chr>      <chr> <chr>   <chr>   <dbl>
## 1 CED217        M     -Inf_0 -Inf       0          15_19 15      19       2803
## 2 CED217        M     -Inf_0 -Inf       0          20_24 20      24        492
## 3 CED217        M     -Inf_0 -Inf       0          25_34 25      34        243
## 4 CED217        M     -Inf_0 -Inf       0          35_44 35      44        218
## 5 CED217        M     -Inf_0 -Inf       0          45_54 45      54        324
## 6 CED217        M     -Inf_0 -Inf       0          55_64 55      64        395
```

```r
head(VIC_tidy)
```

```
## # A tibble: 6 x 9
##   STE_CODE_2021 sex   income income_min income_max age   age_min age_max count
##           <dbl> <chr> <chr>  <chr>      <chr>      <chr> <chr>   <chr>   <dbl>
## 1             2 M     -Inf_0 -Inf       0          15_19 15      19      88386
## 2             2 M     -Inf_0 -Inf       0          20_24 20      24      21186
## 3             2 M     -Inf_0 -Inf       0          25_34 25      34      17702
## 4             2 M     -Inf_0 -Inf       0          35_44 35      44      12908
## 5             2 M     -Inf_0 -Inf       0          45_54 45      54      13821
```

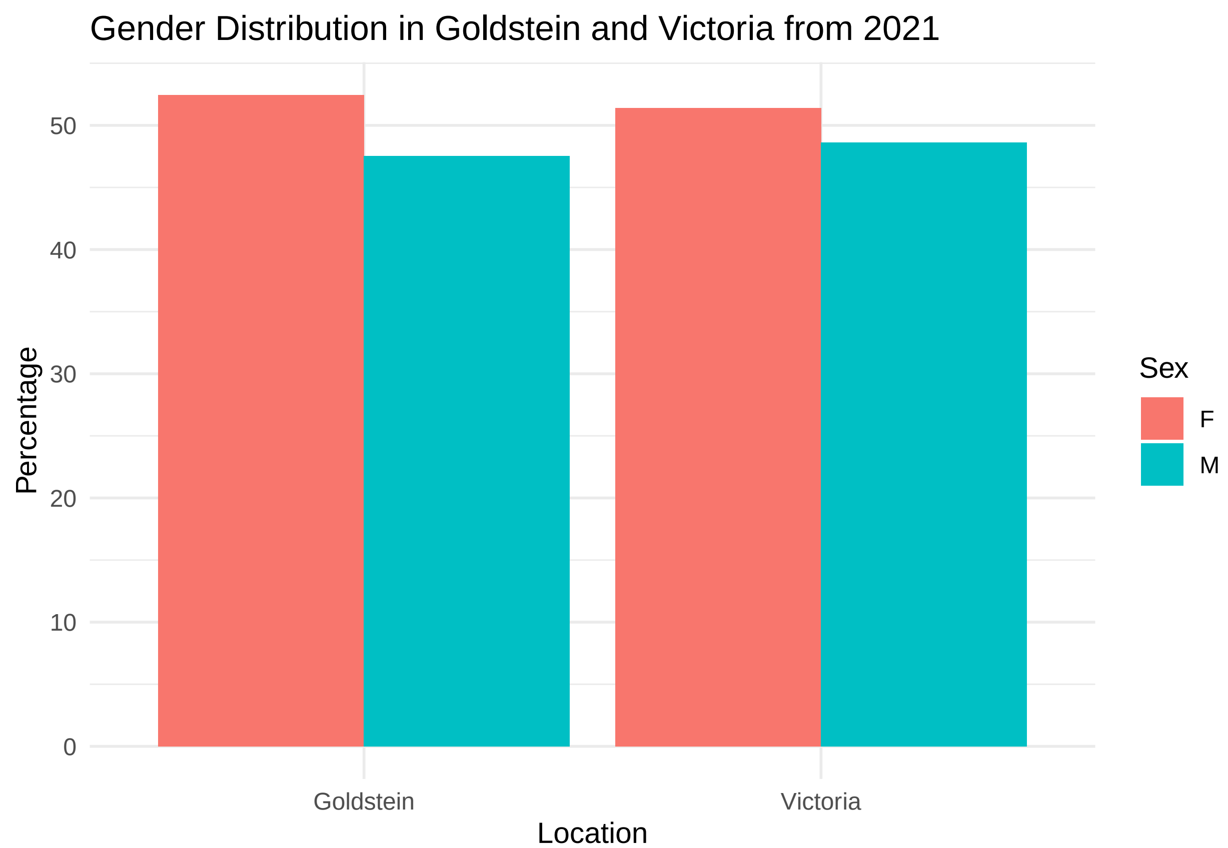
```
## 6             2 M      -Inf_0 -Inf       0          55_64 55      64      20775
```

It should be noted that, the STE code filtering is not necessary in case of STE data, because it contains only data about Victoria.

```
GD_GST <- CED_G17_tidy  |>
  filter(sex != "P")  |>
  group_by(sex)  |>
  summarise(count = sum(count, na.rm = TRUE))

GD_VIC <- VIC_tidy  |>
  filter(sex != "P")  |>
  group_by(sex)  |>
  summarise(count = sum(count, na.rm = TRUE))


GD_GST$location<- "Goldstein" # Creating new column location
GD_VIC$location <- "Victoria"
GD_combined <- rbind(GD_GST, GD_VIC)

# A new column percentage is calculated as both values are not in scale
GD_combined <- GD_combined |>
  group_by(location) |>
  mutate(percentage = count / sum(count) * 100)

GD_combined
```

```
## # A tibble: 4 x 4
## # Groups:   location [2]
##   sex     count location  percentage
##   <chr>   <dbl> <chr>          <dbl>
## 1 F       63584 Goldstein       52.5
## 2 M       57616 Goldstein       47.5
## 3 F     2555888 Victoria        51.4
## 4 M     2417919 Victoria        48.6
```

```
Gender_Dist <- ggplot(GD_combined ,
                      aes(x = location,
                          y = percentage,
                          fill = sex)) +
  geom_col(position = "dodge") +  # Use dodge position for side-by-side bars
  labs(title = "Gender Distribution in Goldstein and Victoria from 2021",
       x = "Location",
       y = "Percentage",
       fill = "Sex") +
  theme_minimal()
Gender_Dist
```



Observations (Gender Ditribution)

- In Victoria, there is a greater proportion of females than males, as depicted by the taller bar representing females compared to males.
- The same can be observed for Goldstein, but with higher proportion.

B)Age Distribution:

```
AD_GST <- CED_G17_tidy  |>
  group_by(age)  |>
  summarise(count = sum(count, na.rm = TRUE))

AD_VIC <-  VIC_tidy  |>
  group_by(age)  |>
  summarise(count = sum(count, na.rm = TRUE))

AD_GST$location<- "Goldstein" # Creating new column location
AD_VIC$location <- "Victoria"
AD_combined <- rbind(AD_GST, AD_VIC)
AD_combined
```

```
## # A tibble: 18 x 3
##    age      count location
##    <chr>    <dbl> <chr>
##  1 15_19    19745 Goldstein
##  2 20_24    16044 Goldstein
##  3 25_34    28454 Goldstein
```

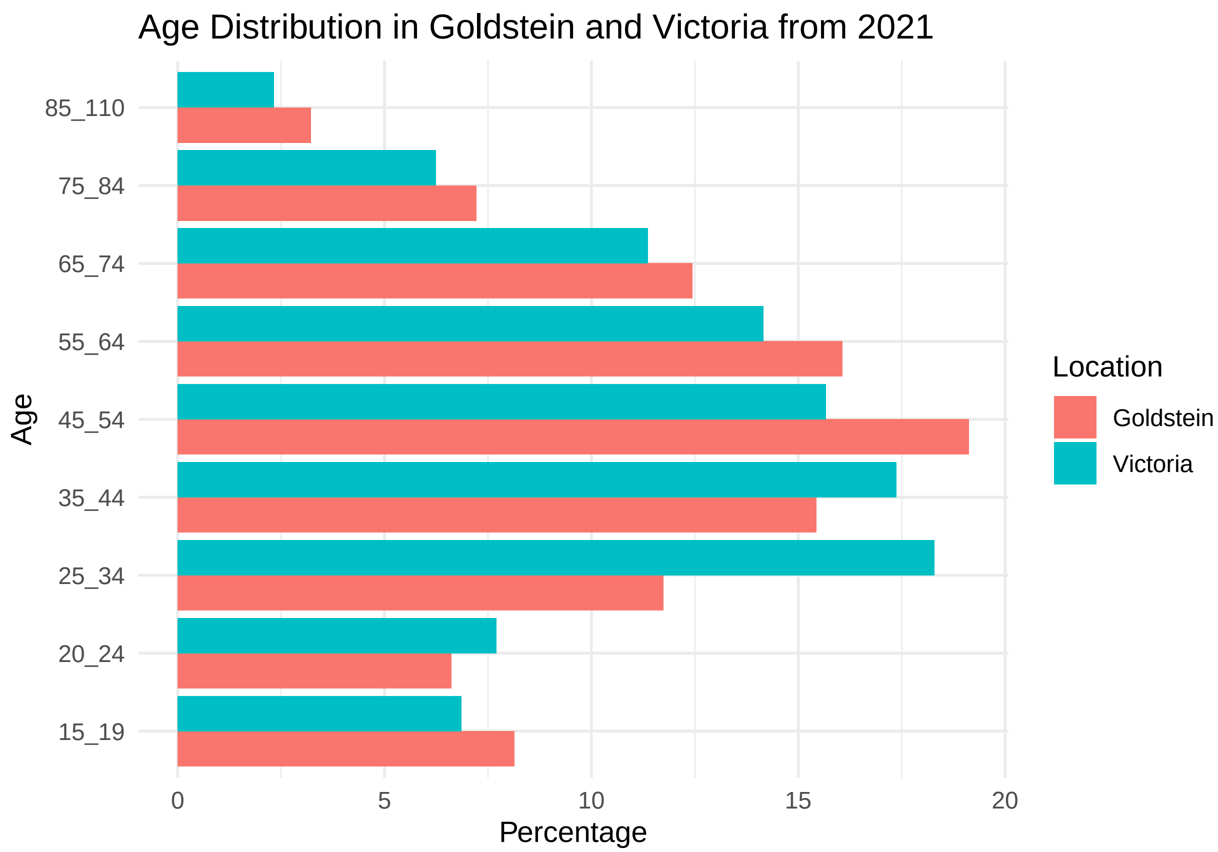
```
##   4 35_44    37416 Goldstein
##   5 45_54    46352 Goldstein
##   6 55_64    38939 Goldstein
##   7 65_74    30148 Goldstein
##   8 75_84    17504 Goldstein
##   9 85_110    7805 Goldstein
##  10 15_19   682200 Victoria
##  11 20_24   766505 Victoria
##  12 25_34  1819795 Victoria
##  13 35_44  1728367 Victoria
##  14 45_54  1558151 Victoria
##  15 55_64  1408746 Victoria
##  16 65_74  1131011 Victoria
##  17 75_84   621174 Victoria
##  18 85_110  231653 Victoria
```

```r
AD_combined <- AD_combined |>
  group_by(location) |>
  mutate(percentage = count / sum(count) * 100)

AD_combined
```

```
## # A tibble: 18 x 4
## # Groups:   location [2]
##    age      count location  percentage
##    <chr>    <dbl> <chr>          <dbl>
##  1 15_19    19745 Goldstein       8.15
##  2 20_24    16044 Goldstein       6.62
##  3 25_34    28454 Goldstein      11.7
##  4 35_44    37416 Goldstein      15.4
##  5 45_54    46352 Goldstein      19.1
##  6 55_64    38939 Goldstein      16.1
##  7 65_74    30148 Goldstein      12.4
##  8 75_84    17504 Goldstein       7.22
##  9 85_110    7805 Goldstein       3.22
## 10 15_19   682200 Victoria        6.86
## 11 20_24   766505 Victoria        7.71
## 12 25_34  1819795 Victoria       18.3
## 13 35_44  1728367 Victoria       17.4
## 14 45_54  1558151 Victoria       15.7
## 15 55_64  1408746 Victoria       14.2
## 16 65_74  1131011 Victoria       11.4
## 17 75_84   621174 Victoria        6.24
## 18 85_110  231653 Victoria        2.33
```

```r
AD_Dist <- ggplot(AD_combined ,
                      aes(x = percentage,
                          y = age,
                          fill = location)) +
  geom_col(position = "dodge") +  # Use dodge position for side-by-side bars
  labs(title = "Age Distribution in Goldstein and Victoria from 2021",
       x = "Percentage",
       y = "Age",
       fill = "Location") +
  theme_minimal()
```

```
AD_Dist
```



Observations (Age Ditribution)

- The age group with **highest population** in **Goldstein** is between **45 to 54 years old**, in contrary the **Victoria** has highest population between **25 to 34 years old**.
- Both young, between 15 and 19 years old, and old population, between 85 and 110 years old, are higher in Goldstein compared to Victoria.

C)Income Distribution:

```
ID_GST <- CED_G17_tidy  |>
  group_by(income)  |>
  summarise(count = sum(count, na.rm = TRUE))  |>
  arrange(desc(count))

ID_VIC <- VIC_tidy  |>
  group_by(income)  |>
  summarise(count = sum(count, na.rm = TRUE))  |>
  arrange(desc(count))

ID_GST$location<- "Goldstein" # Creating new column location
ID_VIC$location <- "Victoria"
ID_combined <- rbind(ID_GST, ID_VIC)
ID_combined
```

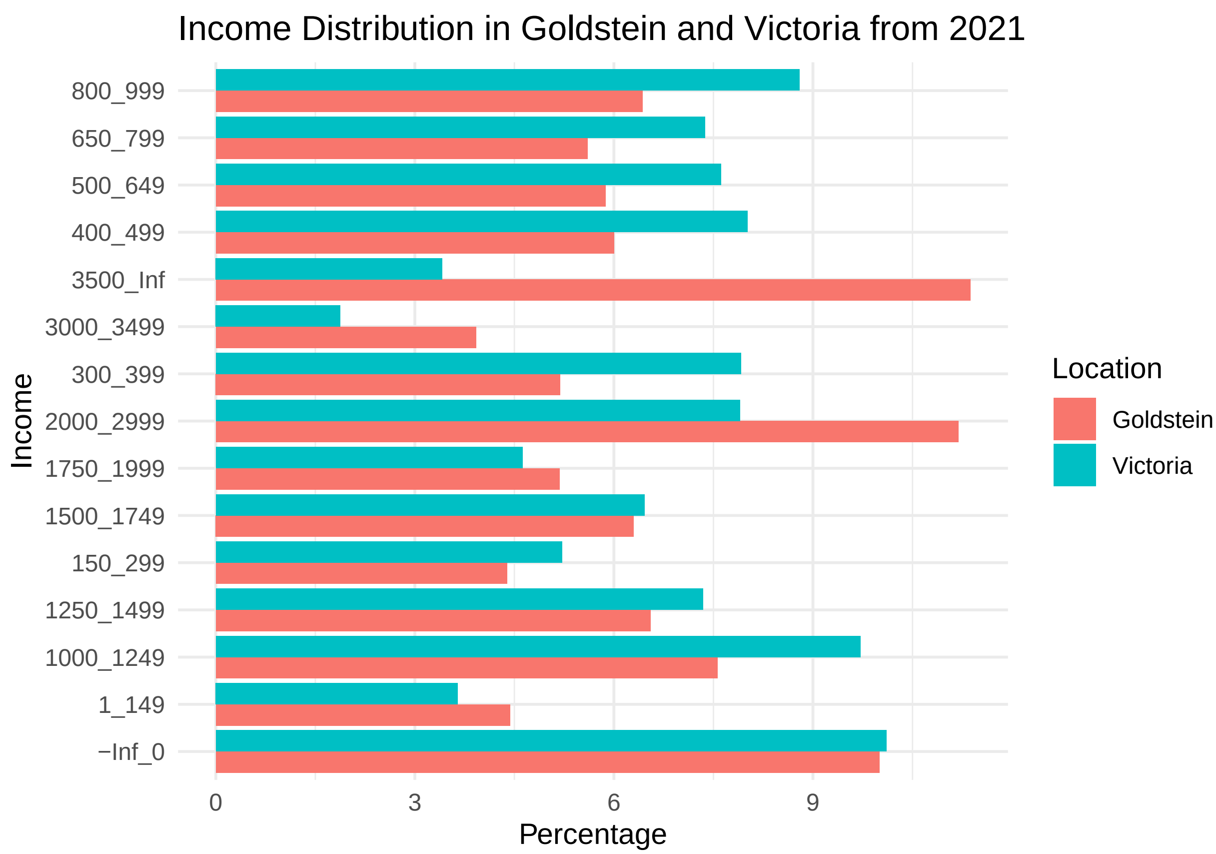
```
## # A tibble: 30 x 3
```

```
##    income    count location
##    <chr>     <dbl> <chr>
##  1 3500_Inf  27563 Goldstein
##  2 2000_2999 27126 Goldstein
##  3 -Inf_0    24247 Goldstein
##  4 1000_1249 18332 Goldstein
##  5 1250_1499 15880 Goldstein
##  6 800_999   15586 Goldstein
##  7 1500_1749 15269 Goldstein
##  8 400_499   14545 Goldstein
##  9 500_649   14238 Goldstein
## 10 650_799   13578 Goldstein
## # i 20 more rows
```

```r
ID_combined <- ID_combined |>
  group_by(location) |>
  mutate(percentage = count / sum(count) * 100)

ID_combined
```

```
## # A tibble: 30 x 4
## # Groups:   location [2]
##    income    count location  percentage
##    <chr>     <dbl> <chr>          <dbl>
##  1 3500_Inf  27563 Goldstein      11.4
##  2 2000_2999 27126 Goldstein      11.2
##  3 -Inf_0    24247 Goldstein      10.0
##  4 1000_1249 18332 Goldstein       7.56
##  5 1250_1499 15880 Goldstein       6.55
##  6 800_999   15586 Goldstein       6.43
##  7 1500_1749 15269 Goldstein       6.30
##  8 400_499   14545 Goldstein       6.00
##  9 500_649   14238 Goldstein       5.87
## 10 650_799   13578 Goldstein       5.60
## # i 20 more rows
```

```r
ID_Dist <- ggplot(ID_combined ,
                       aes(x = percentage,
                           y = income,
                           fill = location)) +
  geom_col(position = "dodge") +  # Use dodge position for side-by-side bars
  labs(title = "Income Distribution in Goldstein and Victoria from 2021",
       x = "Percentage",
       y = "Income",
       fill = "Location") +
  theme_minimal()
ID_Dist
```

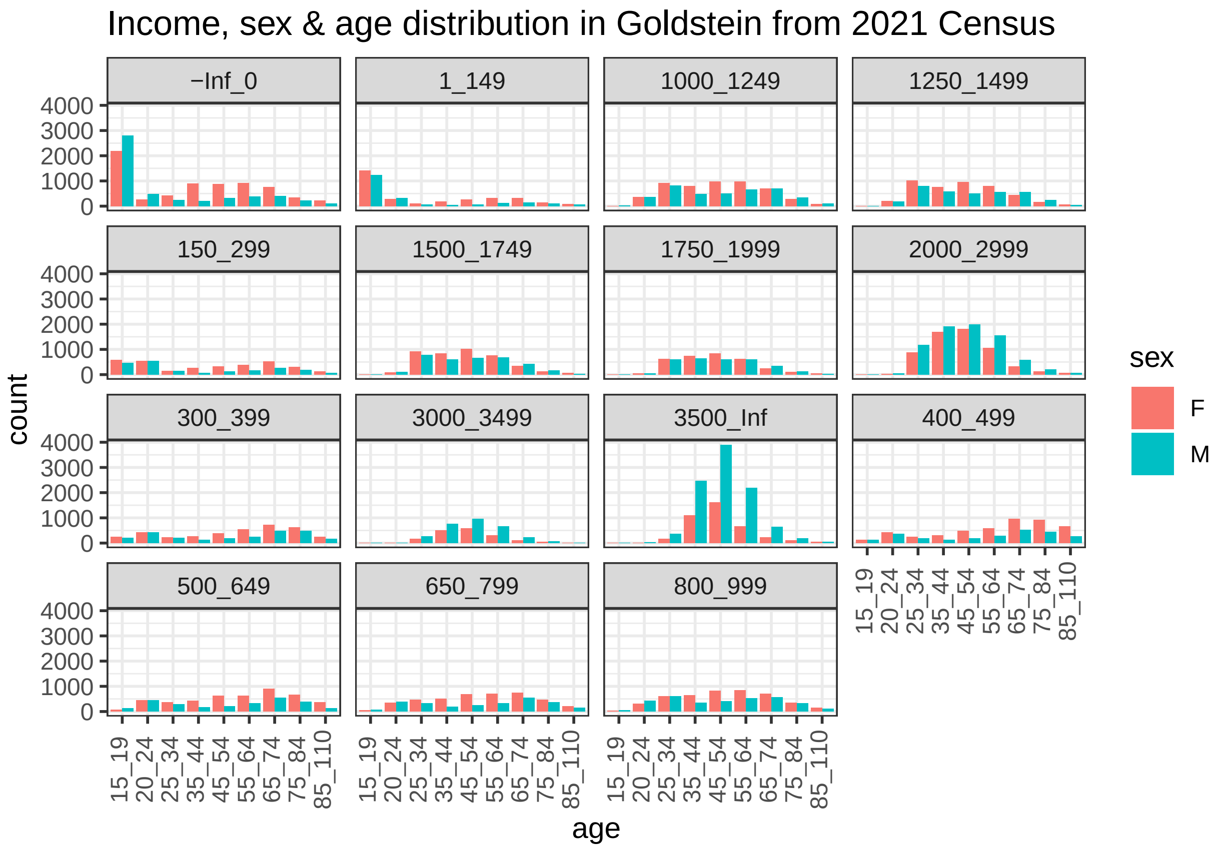


Observations (Income Distribution)

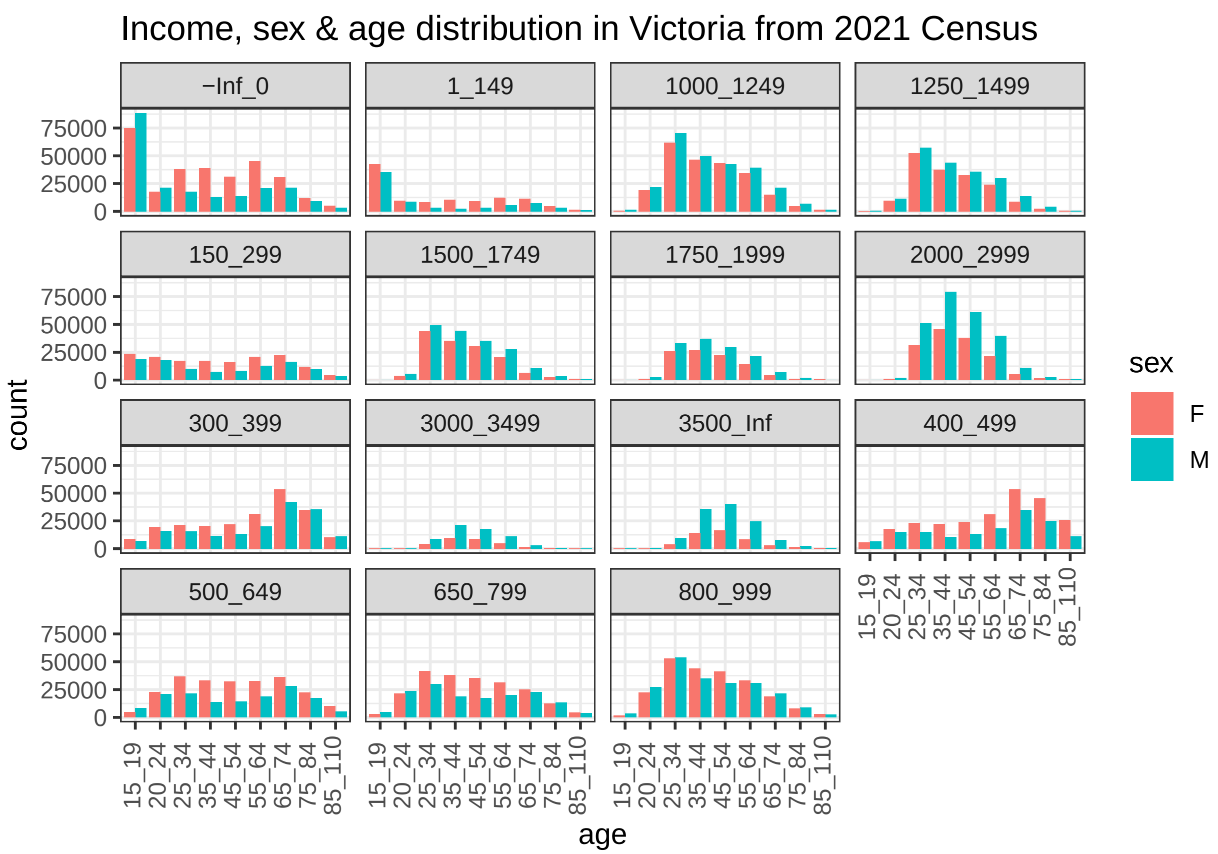
- In Victoria, it looks like people earn a wider range of incomes. The bars on the chart for incomes in Victoria are more spread out than the ones for Goldstein, but in Goldstein, it seems like more people are in a similar income range.
- A kind of similar evenly distributed pattern can be observed in Goldstein. Although, there are more high income earners in Goldstein than Victoria with respect to their corresponding population.

C)Income, sex & age distribution:

```
income_sex_age_distribution_GST <- CED_G17_tidy |>
  filter(sex != "P") |>
  ggplot(aes(x = age, y = count, fill = sex)) +
  geom_col(position = "dodge") +
  facet_wrap(~income) +
  theme_bw(base_size = 11) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.3)) +
  ggtitle("Income, sex & age distribution in Goldstein from 2021 Census")
income_sex_age_distribution_GST
```




```
income_sex_age_distribution_VIC <- VIC_tidy |>
  filter(sex != "P") |>
  ggplot(aes(x = age, y = count, fill = sex)) +
  geom_col(position = "dodge") +
  facet_wrap(~income) +
  theme_bw(base_size = 11) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.3)) +
  ggtitle("Income, sex & age distribution in Victoria from 2021 Census")
income_sex_age_distribution_VIC
```



Overall Observations

- Goldstein looks to be more well-spread when it comes to incomes across age groups, with more men in the high-income ranges and a noticeable number of women in the lower income brackets.
- In contrast, Victoria's wages follow a more straightforward pattern of rising from 25 to 34 years of age and later falling with age. The figure for men in Victoria is in the $2000-$3000 range, contrasting with Goldstein where we have more men earning above $3500.
- In both fields women mainly concentrate in lower income brackets; however, the specific age groups and income levels differ between Goldstein and Victoria.

Median Weekly Rent and Median Total Personal Income Analysis

The geopackages are available in the Australian Bureau of Statistics, ABS.

GeoPackages contain comprehensive data about people, families and dwellings for Census geographical areas ranging from Statistical Areas Level 1 (SA1) to the whole of Australia.

GeoPackages are Census data linked to geographical boundary data from the **Australian Statistical Geography Standard (ASGS)** in a SQLite container.

To navigate to the download page of geopackages the path is as below:

Australian Bureau of Statistics > Census > Find Census data > Census data tools > GeoPacks

The following attributes are set:

- Census year - 2021

- State or territory - Victoria
- with table options - G02 ~ Selected medians and averages
- GDA type - GDA2020

Initially the Geopackage is loaded and the layers are checked.

```
gpkg_file <- "data/Geopackage_2021_G02_VIC_GDA2020/G02_VIC_GDA2020.gpkg"

gpkg_layers <- st_layers(gpkg_file)
gpkg_layers
```

```
## Driver: GPKG
## Available layers:
##            layer_name geometry_type features fields crs_name
## 1    G02_UCL_2021_VIC Multi Polygon      347     11  GDA2020
## 2    G02_SUA_2021_VIC Multi Polygon       22     11  GDA2020
## 3    G02_STE_2021_VIC Multi Polygon        1     11  GDA2020
## 4   G02_SOSR_2021_VIC Multi Polygon       12     11  GDA2020
## 5    G02_SOS_2021_VIC Multi Polygon        6     11  GDA2020
## 6    G02_SED_2021_VIC Multi Polygon       90     11  GDA2020
## 7    G02_SAL_2021_VIC Multi Polygon     2946     11  GDA2020
## 8    G02_SA4_2021_VIC Multi Polygon       19     11  GDA2020
## 9    G02_SA3_2021_VIC Multi Polygon       68     11  GDA2020
## 10   G02_SA2_2021_VIC Multi Polygon      524     11  GDA2020
## 11   G02_SA1_2021_VIC Multi Polygon    15482     11  GDA2020
## 12   G02_POA_2021_VIC Multi Polygon      694     11  GDA2020
## 13   G02_LGA_2021_VIC Multi Polygon       82     11  GDA2020
## 14 G02_GCCSA_2021_VIC Multi Polygon        4     11  GDA2020
## 15   G02_CED_2021_VIC Multi Polygon       41     11  GDA2020
## 16    G02_RA_2021_VIC Multi Polygon        6     11  GDA2020
```

The layer "G02_CED_2021_VIC" is selected from which two variables has to be chosen.

```
vicmap_sa1_G02 <- read_sf("data/Geopackage_2021_G02_VIC_GDA2020/G02_VIC_GDA2020.gpkg", layer = "G02_SA1_
vicmap_sa1_G02
```

```
## Simple feature collection with 15482 features and 11 fields (with 4 geometries empty)
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 141 ymin: -39.2 xmax: 150 ymax: -34
## Geodetic CRS:  GDA2020
## # A tibble: 15,482 x 12
##    SA1_CODE_2021 SA1_NAME_2021 Median_age_persons Median_mortgage_repay_monthly
##    <chr>         <chr>                      <dbl>                         <dbl>
##  1 20101100101   20101100101                   39                          1452
##  2 20101100102   20101100102                   49                          1257
##  3 20101100105   20101100105                   33                          1136
##  4 20101100106   20101100106                   40                          1302
##  5 20101100107   20101100107                   46                          1044
##  6 20101100108   20101100108                   35                          1517
##  7 20101100109   20101100109                   39                          1300
##  8 20101100110   20101100110                   45                          1647
##  9 20101100111   20101100111                   40                          1733
## 10 20101100112   20101100112                   47                          1517
## # i 15,472 more rows
## # i 8 more variables: Median_tot_prsnl_inc_weekly <dbl>,
```

```
## #   Median_rent_weekly <dbl>, Median_tot_fam_inc_weekly <dbl>,
## #   Average_num_psns_per_bedroom <dbl>, Median_tot_hhd_inc_weekly <dbl>,
## #   Average_household_size <dbl>, AREA_ALBERS_SQKM <dbl>,
## #   geom <MULTIPOLYGON [°]>
```

```
names(vicmap_sa1_G02)
```

```
##  [1] "SA1_CODE_2021"                "SA1_NAME_2021"
##  [3] "Median_age_persons"           "Median_mortgage_repay_monthly"
##  [5] "Median_tot_prsnl_inc_weekly"  "Median_rent_weekly"
##  [7] "Median_tot_fam_inc_weekly"    "Average_num_psns_per_bedroom"
##  [9] "Median_tot_hhd_inc_weekly"    "Average_household_size"
## [11] "AREA_ALBERS_SQKM"             "geom"
```

The two variables which are beneficial in terms of politics and election can be **"Median__age__persons"** and **"Median__tot__prsnl__inc__weekly"**.

Median rent per week:

- Gives the median weekly rent a person pays for the stay.
- Rent prices can illustrates the affordability of housing. It can also indicate the composition of the population(renters and homeowners)
- This can influence political preferences of the demographics.
- As for the huge rents – they are number one enemy for the tenants. This means that people will tend to vote for those that want to make sure that the rent is affordable.
- Parties could get more votes by promising to assist on accommodation expenditures. However, the parties not talking about rent issues might lose support from the electorate.

Median Total Personal Income Weekly:

- Gives the median total personal income per week.
- Income levels often has correlation with the outcome of an election, where wealthy individuals have different grievances compared to the middle class and poor individuals.
- For example, Lower income areas might lean on policies supporting social welfare, while higher income areas might want tax policies or business incentives.

Spatial data:

Spatial data on Federal electorate boundary can be found in Federal electoral boundary GIS data in the official Australian Electoral Commision website.

After agreeing to the license we will be directed to the download page where we choose the spatial data of Victoria, for this analysis ".shp" format is downloaded and used.

```
aec_map <- read_sf(here::here("data/vic-july-2021-esri/E_VIC21_region.shp"))
```

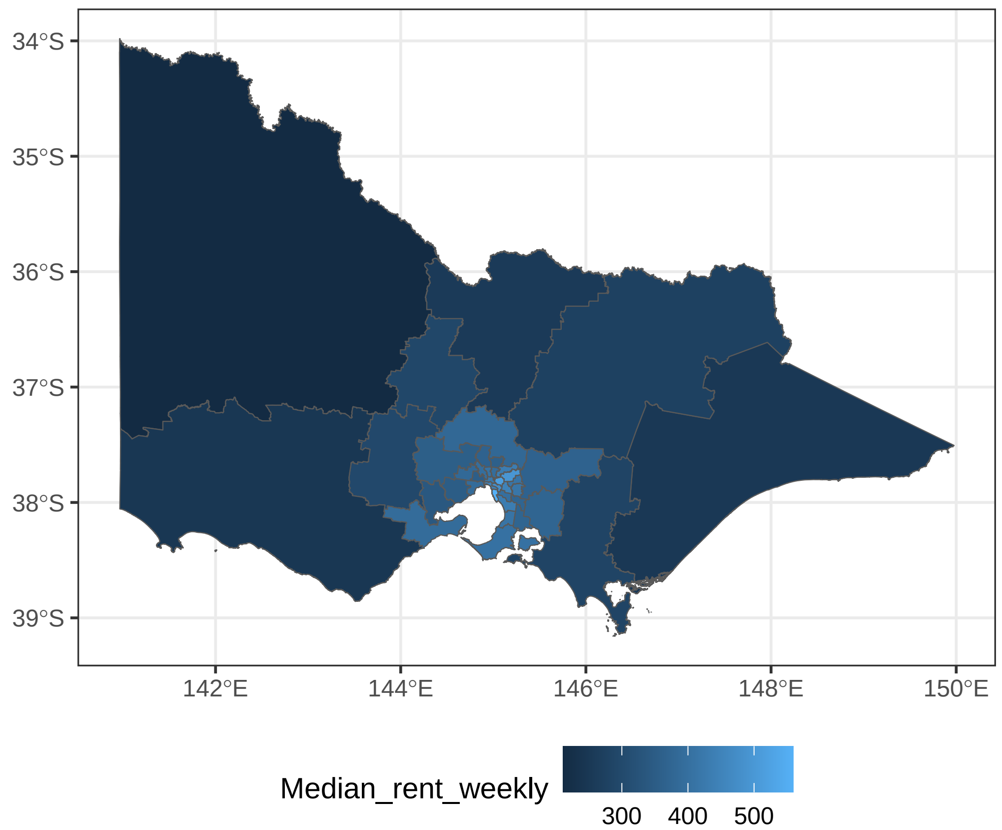
In this analysis we are choosing two variables which equals to two graphs, to avoid repeating the codes a function can be created.

```
# Define a function to process map data
process_map_data <- function(map_data, property, coords, divisions) {
  # Set projection to GDA1994 using EPSG:4283
  st_crs(map_data$geometry, 4283)

  # Transform projection from GDA1994 to GDA2020 using EPSG:7844
  map_data$geometry = st_transform(map_data$geometry, 7844)
```

```
# Get SA1 Centroids
vicmap_sa1_G02_w_centroid <- vicmap_sa1_G02 |>
  mutate(centroid = st_centroid(geom))

# Determine which SA1 centroids overlap with which electoral regions
electoral_intersects = st_intersects(vicmap_sa1_G02_w_centroid$centroid,
                                     map_data$geometry,
                                     sparse = FALSE)
arr_ind = which(electoral_intersects == TRUE, arr.ind = TRUE)
sa1_ind = arr_ind[,1]
division_ind = arr_ind[,2]
division_name = map_data[[divisions]][division_ind]
sa1_name = vicmap_sa1_G02_w_centroid$SA1_NAME_2021[sa1_ind]

# Save the SA1 region and its matching electorate into a data frame
sa1_divisions = data.frame(SA1_NAME_2021 = sa1_name,
                           DivisionNm = division_name)

# Add the electorate names to the map data
vicmap_sa1_G02_electorates <- vicmap_sa1_G02 |>
  right_join(sa1_divisions)

# Combine the data from electorates together using group_by
grouped_data <- vicmap_sa1_G02_electorates |>
  group_by(DivisionNm) |>
  summarise_at(vars({{property}}), ~mean(., na.rm = TRUE)) |>
  ungroup()

# Plot results to check assignment
map_plot <- ggplot() +
  geom_sf(data = grouped_data,
          aes(geometry = geom, fill = {{property}})) +
  theme_bw() +
  theme(legend.position = "bottom")

# set the coordinate limits for the chosen electorates
cropped_map <- grouped_data |>
  st_crop(xmin = coords[1], xmax = coords[2],
          ymin = coords[3], ymax = coords[4])

cropped_plot <- ggplot(cropped_map) +
  geom_sf(aes(geometry = geom, fill = {{property}})) +
   theme(
    legend.position = "bottom",
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 8),
    plot.title = element_text(size = 12),
    axis.title = element_blank()
  )

# Labelling the Map
data_for_labels <- cropped_map |>
  filter(DivisionNm %in% c("Kooyong", "Goldstein", "Melbourne", "Menzies", "Maribyrnong", "Wills", "S
```

```
  labelled_plot <- cropped_plot +
    geom_sf_label(data = data_for_labels,
                  aes(label = DivisionNm, geometry = geom),
                  size = 2)

  list(map_plot = map_plot, cropped_plot = cropped_plot, labelled_plot = labelled_plot)


}

# Usage for Map MWR
map_MWR <- process_map_data(aec_map, Median_rent_weekly, c(144.8, 145.2, -38.1, -37.6), "Elect_div")

# Usage for Map MTPIW
map_MTPIW <- process_map_data(aec_map, Median_tot_prsnl_inc_weekly, c(144.8, 145.2, -38.1, -37.6), "Ele
```

```
map_MWR$map_plot
```



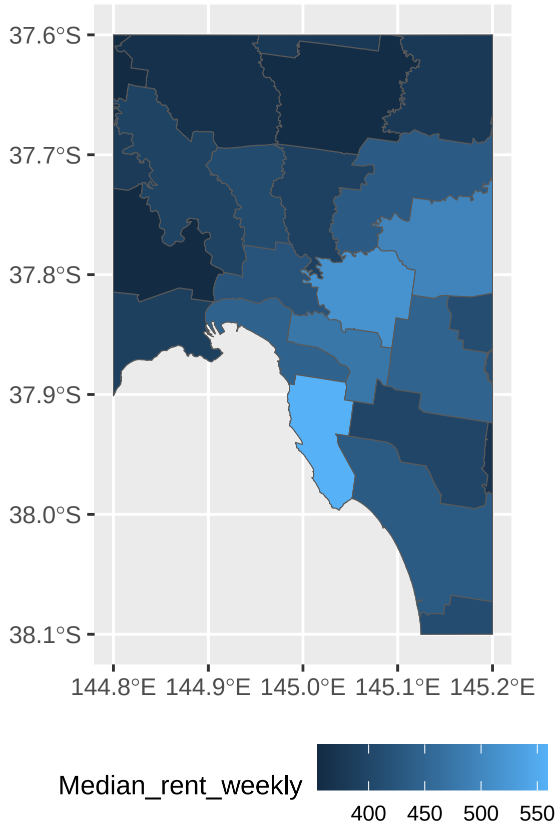
Median rent per week

Map of the entire state of Victoria, with its median weekly rent, is not a good visualization to draw conclusions from, because the much smaller locations are tiny and does not illustrates the varying median weekly rents with the color gradients.

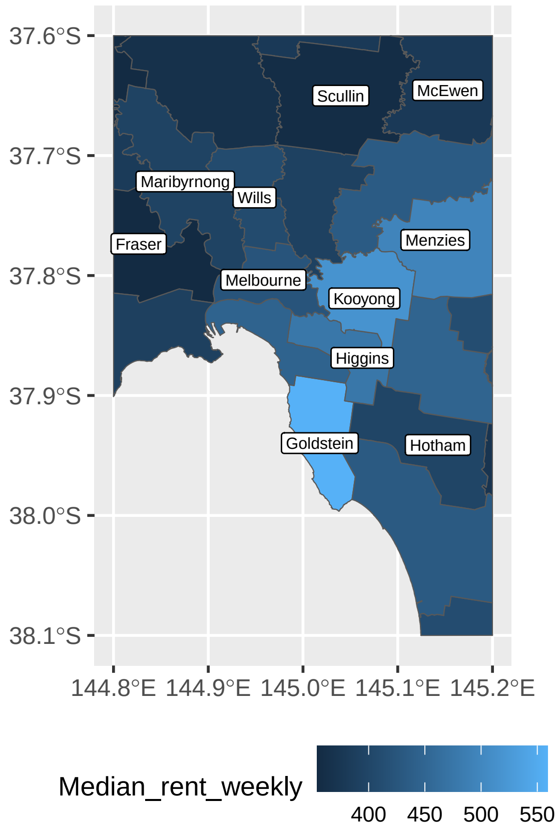
As a result a scoped view would give us more clear picture.

For this we choose certain electorates and set the coordinate limits of the maps.

```
map_MWR$cropped_plot
```



Now that we have the scoped view of the map, it's important to note that a map includes the names of locations. Similarly, we label the locations on the map as follows:



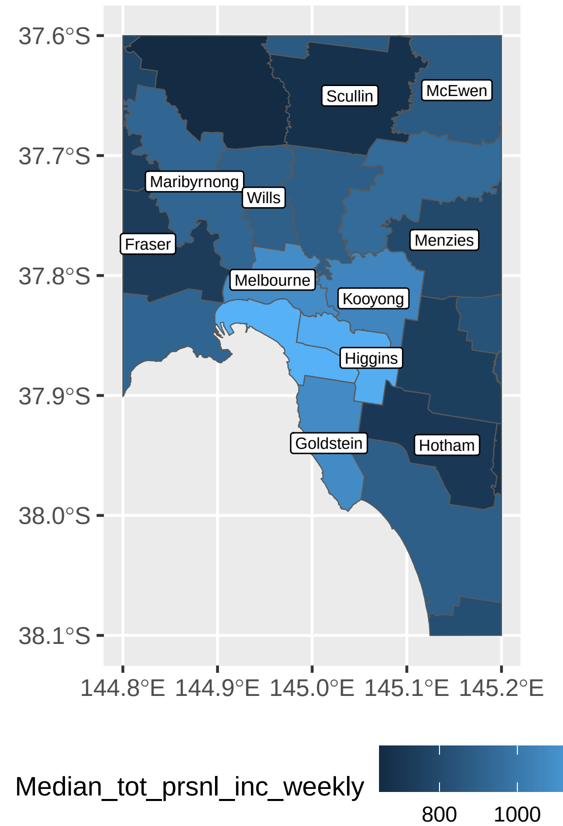
Observations on Median weekly rent Rent Differences Across Electorates:

- In Goldstein County, the median rent is over AUD 550 per week. Kooyong, with rents exceeding AUD 500 per week.
- In Menzies County, the median rent is nearly AUD 450 per week.

Urban vs. Suburban Rent Trends:

- We observe a trend where median rent decreases as we move from urban to suburban areas.
- Suburbs like Scullin, Wills, and Marribyrnong have notably lower weekly rents compared to the urban areas, illustrating more affordable housing options in these areas.

```
map_MTPIW$labelled_plot
```



Median Total Personal Income Weekly

Observations on Median total personal income per week

- Regions such as Goldstein and Kooyong exhibit a higher median income, indicating that residents in these areas tend to earn more compared to those in other regions.
- Even though Kooyong and Goldstein are wealthy regions based on median income, they were won by independent candidates. This suggests that despite residents earning more on average, they chose independent candidates who focused on local concerns or transparency, rather than sticking to traditional party choices.
- Different job sectors offer different pay scales. Regions with more finance, tech, or professional services jobs often have higher median incomes compared to areas with more manufacturing, retail, or hospitality jobs.
- Wealthy people can afford to live in so called high cost-of-living locations, which implicates on high median income level.

Referendum

A referendum is conducted when citizens believe there should be alterations to their current constitution. Changes are implemented if the majority of voters support “Yes”.

How it works:

1. A Bill is passed by Parliament

Before a referendum can be held, a bill outlining the proposed changes to the constitution must be passed by both houses of the Federal Parliament, or alternatively passed twice in either the House of Representatives or the Senate.

2. A Writ is issued

The Governor-General issues a writ for a referendum which, like an election must be held on a Saturday. It can be held with an ordinary election but can also be held separately.

3. Australians vote

The earliest a referendum can occur is the first Saturday falling two months and 33 days after the bill is passed. The latest is the final Saturday falling before six months has elapsed since the bill was passed.

There can be several proposed changes on a referendum ballot paper for voters to consider. If voters agree with a proposed change, they write 'yes' in the square on the ballot paper. If they do not agree with a proposed change, they write 'no' in the square.

Data acquisition:

You can find data on previously held referendums in Australia on the Australian Electoral Commission (AEC) website. To access the information for the 2023 Referendum, follow these steps:

1.Go to the AEC website:

2.Navigate to the "Referendums" from the Information Centre dropdown menu.

3.Look for the section titled "Previous Referendums" and click on it.

4.Within the list of previous referendums, locate and click on the "2023 Referendum.

Once you are on the page for the 2023 Referendum, you will find a section titled "Downloads and Statistics" under the "Supporting Information" tab. Click on this section to access the raw data , under "Results by vote type", for the referendum results.

```
url = "https://www.aec.gov.au/Elections/referendums/files/ReferendumResultsByVoteType.csv"
ref_data <- read_csv(url)
ref_data
```

```
## # A tibble: 151 x 33
##    `Division ID` Division  State `Ordinary Yes Votes` `Ordinary Yes % of Formal`
##            <dbl> <chr>     <chr>                <dbl>                      <dbl>
##  1           179 Adelaide  SA                   44215                       50.4
##  2           197 Aston     VIC                  34646                       42.5
##  3           198 Ballarat  VIC                  36411                       42.2
##  4           103 Banks     NSW                  31297                       38.8
##  5           180 Barker    SA                   20600                       21.6
##  6           104 Barton    NSW                  34112                       44.0
##  7           192 Bass      TAS                  22570                       39.0
##  8           318 Bean      ACT                  49487                       56.4
##  9           200 Bendigo   VIC                  35810                       40.7
## 10           105 Bennelong NSW                  41315                       48.9
## # i 141 more rows
## # i 28 more variables: `Ordinary No Votes` <dbl>,
## #   `Ordinary No % of Formal` <dbl>, `Ordinary Informal Votes` <dbl>,
## #   `Ordinary Informal % of Total` <dbl>, `Absent Yes Votes` <dbl>,
## #   `Absent Yes % of Formal` <dbl>, `Absent No Votes` <dbl>,
## #   `Absent No % of Formal` <dbl>, `Absent Informal Votes` <dbl>,
## #   `Absent Informal % of Total` <dbl>, `Provisional Yes Votes` <dbl>, ...
```

We will use the `janitor` package to clean up the variable names in the dataset for better readability.

```
ref_data_clean <- ref_data |> clean_names()

names <- names(ref_data_clean)
head(names, 10)
```

```
##  [1] "division_id"                        "division"
##  [3] "state"                              "ordinary_yes_votes"
##  [5] "ordinary_yes_percent_of_formal"     "ordinary_no_votes"
##  [7] "ordinary_no_percent_of_formal"      "ordinary_informal_votes"
##  [9] "ordinary_informal_percent_of_total" "absent_yes_votes"
```

```
names(ref_data_clean)
```

```
##  [1] "division_id"
##  [2] "division"
##  [3] "state"
##  [4] "ordinary_yes_votes"
##  [5] "ordinary_yes_percent_of_formal"
##  [6] "ordinary_no_votes"
##  [7] "ordinary_no_percent_of_formal"
##  [8] "ordinary_informal_votes"
##  [9] "ordinary_informal_percent_of_total"
## [10] "absent_yes_votes"
## [11] "absent_yes_percent_of_formal"
## [12] "absent_no_votes"
## [13] "absent_no_percent_of_formal"
## [14] "absent_informal_votes"
## [15] "absent_informal_percent_of_total"
## [16] "provisional_yes_votes"
## [17] "provisional_yes_percent_of_formal"
## [18] "provisional_no_votes"
## [19] "provisional_no_percent_of_formal"
## [20] "provisional_informal_votes"
## [21] "provisional_informal_percent_of_total"
## [22] "pre_poll_yes_votes"
## [23] "pre_poll_yes_percent_of_formal"
## [24] "pre_poll_no_votes"
## [25] "pre_poll_no_percent_of_formal"
## [26] "pre_poll_informal_votes"
## [27] "pre_poll_informal_percent_of_total"
## [28] "postal_yes_votes"
## [29] "postal_yes_percent_of_formal"
## [30] "postal_no_votes"
## [31] "postal_no_percent_of_formal"
## [32] "postal_informal_votes"
## [33] "postal_informal_percent_of_total"
```

We have filtered the dataset to include observations only for the chosen electorate, Goldstein.

```
ref_data_GST <- ref_data_clean |>
  filter(division == "Goldstein")

# Calculate the total "Yes" votes for each row
ref_data_GST <- ref_data_GST |>
  mutate(total_yes = sum(ordinary_yes_votes, absent_yes_votes,
```

```r
                         provisional_yes_votes, pre_poll_yes_votes,
                         postal_yes_votes))

# Calculate the total votes (Yes + No + Informal) for each row
ref_data_GST <- ref_data_GST |>
  mutate(total_votes = sum(ordinary_yes_votes, absent_yes_votes,
                           provisional_yes_votes, pre_poll_yes_votes,
                           postal_yes_votes, ordinary_no_votes,
                           absent_no_votes, provisional_no_votes,
                           pre_poll_no_votes, postal_no_votes,
                           ordinary_informal_votes, absent_informal_votes,
                           provisional_informal_votes, pre_poll_informal_votes,
                           postal_informal_votes))

# Calculate the percentage of "Yes" votes for each row
ref_data_GST <- ref_data_GST |>
  mutate(yes_percentage = (total_yes / total_votes) * 100)


ref_data_GST$yes_percentage
```

```
## [1] 55.7
```

The **Yes percentage** for Goldstein electorate is **55.7%**

Since this referendum was about adding an Indigenous voice in Parliament, it'd be helpful to know how many people in our chosen area identify as Indigenous or Torres Strait Islander.

To find this information, we revisit the ABS website's datapacks page. We'll choose the "Aboriginal and/or Torres Strait Islander Peoples (Indigenous) Profile" datapack type, set the geography to Victoria, and select the 2021 census year to download the data.

Now explore and search for suitable topic related to finding the Indigenous Demographics or the Indigenous population.

"I02" variable can be beneficial to find the number of indigenous people in the chosen electorate as it contains information on **Indigenous Status by Sex**.

The interesting thing to note in the downloaded zip file is that there is no data with respect to SA1 regions in this folder.

Since it is small geographic area among ABS data, the package that describes Indigenous characteristics is not provided for these regions in order to preserve people privacy. Moreover, the small sample sizes in the communities with a very low population of Indigenous people might cause misinformation to be presented in the representation of those data.

Is my electorate Representative of the broader population

Voter Behavior:

Independent Candidate Wins: Despite higher incomes, Goldstein and Kooyong elected independent candidates.

Observation: Voters in these areas might prioritize local concerns over traditional party affiliations, possibly differing from the state's usual voting patterns.

Socioeconomic Diversity:

Higher Median Incomes: Goldstein and Kooyong have higher incomes.

Observation: This might mean these areas are more affluent, possibly not fully representing the state's diverse socioeconomic groups.

Age Distribution:

Goldstein's Older Population: Goldstein has more residents aged 45 to 54.

Observation: This might not perfectly reflect the state's age groups, impacting Goldstein's representativeness in terms of age.

Gender Imbalance:

More Females in Goldstein: Goldstein has a higher proportion of females.

Observation: This might not match the state's gender distribution, affecting Goldstein's representation.

Housing Affordability:

Higher Median Rents: Goldstein and Kooyong have higher rents.

Observation: This could mean these areas are less affordable for lower-income residents, potentially not fully representing all income levels in the state.

Citations

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, *4*(43), 1686. doi:10.21105/joss.01686 https://doi.org/10.21105/joss.01686.

Wickham H (2023). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 1.0.0, https://CRAN.R-project.org/package=forcats.

Wickham H, Bryan J (2023). *readxl: Read Excel Files*. R package version 1.4.3, https://CRAN.R-project.org/package=readxl.

Pebesma E, Bivand R (2023). *Spatial Data Science: With applications in R*. Chapman and Hall/CRC. doi:10.1201/9780429459016 https://doi.org/10.1201/9780429459016, https://r-spatial.org/book/.

Pebesma E (2018). "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal*, *10*(1), 439-446. doi:10.32614/RJ-2018-009 https://doi.org/10.32614/RJ-2018-009, https://doi.org/10.32614/RJ-2018-009.

Firke S (2023). *janitor: Simple Tools for Examining and Cleaning Dirty Data*. R package version 2.2.0, https://CRAN.R-project.org/package=janitor.

Wickham H, Henry L (2023). *purrr: Functional Programming Tools*. R package version 1.0.2, https://CRAN.R-project.org/package=purrr.


```
## [[1]]
## Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
## Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E,
## Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi
## K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the
## tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
## doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
##
## [[2]]
## Wickham H (2023). _forcats: Tools for Working with Categorical
## Variables (Factors)_. R package version 1.0.0,
## <https://CRAN.R-project.org/package=forcats>.
```

```
##
## [[3]]
## Wickham H, Bryan J (2023). _readxl: Read Excel Files_. R package
## version 1.4.3, <https://CRAN.R-project.org/package=readxl>.
##
## [[4]]
## Pebesma E, Bivand R (2023). _Spatial Data Science: With applications in
## R_. Chapman and Hall/CRC. doi:10.1201/9780429459016
## <https://doi.org/10.1201/9780429459016>, <https://r-spatial.org/book/>.
##
## Pebesma E (2018). "Simple Features for R: Standardized Support for
## Spatial Vector Data." _The R Journal_, *10*(1), 439-446.
## doi:10.32614/RJ-2018-009 <https://doi.org/10.32614/RJ-2018-009>,
## <https://doi.org/10.32614/RJ-2018-009>.
##
## [[5]]
## Firke S (2023). _janitor: Simple Tools for Examining and Cleaning Dirty
## Data_. R package version 2.2.0,
## <https://CRAN.R-project.org/package=janitor>.
##
## [[6]]
## Wickham H, Henry L (2025). _purrr: Functional Programming Tools_. R
## package version 1.0.4, <https://CRAN.R-project.org/package=purrr>.
```

Australian Electoral Commission

Australian Bureau of Statistics

OpenAI. (2022). ChatGPT (Version 3.5) [Computer software]. https://openai.com/

ChatGPT 3.5

You

How to encapsulate all these repeating codes into a function and at last when I call the function with the list of datasets it should give me the final data named CED_G17_tidy_VIC

```
CED_G17A_long <- CED_G17A |>
 pivot_longer(cols = -1, names_to = "classification", # Transforming variables into "classification"
        values_to = "count") # Assigning observations to "counts"

# Converting CED_G17B from wide to long format
CED_G17B_long <- CED_G17B |>
 pivot_longer(cols = -1, names_to = "classification",
        values_to = "count")

# Converting CED_G17C from wide to long format
CED_G17C_long <- CED_G17C |>
 pivot_longer(cols = -1, names_to = "classification",
        values_to = "count")

# Combining all rows to compile the full dataset for plotting
CED_G17_long <- bind_rows(CED_G17A_long, CED_G17B_long, CED_G17C_long)

# Removing "Tot" (totals) and "PI_NS" (personal income not stated) columns
CED_G17_long_processed <- CED_G17_long |>
 filter(!str_detect(string = classification, pattern = "Tot"),
     !str_detect(classification, pattern = "PI_NS"))

# Replacing inappropriate naming conventions using `string_replace`
CED_G17_long_processed <- CED_G17_long_processed |>
 mutate(
  classification = str_replace(classification, "Neg_Nil_income", "-Inf_0"),
  classification = str_replace(classification, "Neg_Nil_incme", "-Inf_0"),
  classification = str_replace(classification, "Negtve_Nil_incme", "-Inf_0"),
  classification = str_replace(classification, "more", "Inf"),
  classification = str_replace(classification, "85ov", "85_110_yrs"),
  classification = str_replace(classification, "85_yrs_ovr", "85_110_yrs"))

# Expanding the dataset to separate different values into each variable
CED_G17_tidy_VIC <- CED_G17_long_processed |>
 mutate(classification = str_remove(classification, "_yrs")) |>
 separate_wider_delim(cols = classification, delim = "_",
            names = c("sex", "income_min", "income_max", "age_min", "age_max"))
```

ChatGPT

You can encapsulate these repetitive operations into a function that takes the dataset as an argument. Here's how you can do it:

```r
library(dplyr)
library(tidyr)

process_dataset <- function(dataset) {
  dataset_long <- dataset |>
    pivot_longer(cols = -1, names_to = "classification", values_to = "count")

  dataset_long_processed <- dataset_long |>
    filter(!str_detect(string = classification, pattern = "Tot"),
           !str_detect(classification, pattern = "PI_NS")) |>
    mutate(
      classification = str_replace(classification, "Neg_Nil_income", "-Inf_0"),
      classification = str_replace(classification, "Neg_Nil_incme", "-Inf_0"),
      classification = str_replace(classification, "Negtve_Nil_incme", "-Inf_0"),
      classification = str_replace(classification, "more", "Inf"),
      classification = str_replace(classification, "85ov", "85_110_yrs"),
      classification = str_replace(classification, "85_yrs_ovr", "85_110_yrs"))

  dataset_tidy <- dataset_long_processed |>
    mutate(classification = str_remove(classification, "_yrs")) |>
    separate_wider_delim(cols = classification, delim = "_",
                         names = c("sex", "income_min", "income_max", "age_min", "age

  return(dataset_tidy)
}

# List of datasets
datasets <- list(CED_G17A, CED_G17B, CED_G17C)

# Apply the function to each dataset and combine them
CED_G17_tidy_VIC <- datasets |>
  lapply(process_dataset) |>
  bind_rows()
```

This function `process_dataset` takes a dataset, performs all the necessary operations, and returns the processed dataset. Then, you can use `lapply` to apply this function to each dataset in your list of datasets and finally bind them together using `bind_rows()` to get the final dataset `CED_G17_tidy_VIC`.

Message ChatGPT...

ChatGPT can make mistakes. Consider checking important information.

Function for Task 3

OpenAI. (2022). ChatGPT (Version 3.5) [Computer software]. https://openai.com/

ChatGPT 3.5

You

I have these repeating codes create a function for it, i am using r language

~~~~
```{r Map data - Task 4}
aec_map <- read_sf(here::here("data/vic-july-2021-esri/E_VIC21_region.shp"))
```
```{r Wrangling data for Map MWR - Task 4}
# Set projection to GDA1994 using EPSG:4283
st_crs(aec_map$geometry,4283)

# Transform projection from GDA1994 to GDA2020 using EPSG:7844
aec_map$geometry = st_transform(aec_map$geometry, 7844)

# Get SA1 Centroids
vicmap_sa1_G02_w_centroid <- vicmap_sa1_G02 |>
  mutate(centroid = st_centroid(geom))

# Determine with SA1 centroids overlap with which electoral regions
electoral_intersects = st_intersects(vicmap_sa1_G02_w_centroid$centroid,
                      aec_map$geometry,
                      sparse = FALSE)
arr_ind = which(electoral_intersects == TRUE, arr.ind = TRUE)
sa1_ind = arr_ind[,1]
division_ind = arr_ind[,2]
division_name = aec_map$Elect_div[division_ind]
sa1_name = vicmap_sa1_G02_w_centroid$SA1_NAME_2021[sa1_ind]

# Save the SA1 region and its matching electorate into a data frame
sa1_divisions = data.frame(SA1_NAME_2021 = sa1_name,
                    DivisionNm = division_name)

# Add the electorate names to the map
vicmap_sa1_G02_electorates  <- vicmap_sa1_G02 %>%
  right_join(sa1_divisions)

# Combine the data from electorates together using group_by
vicmap_sa1_G02_electorates_grouped_MWR =  vicmap_sa1_G02_electorates |>
  group_by(DivisionNm) |>
  summarise(Median_rent_weekly = mean(Median_rent_weekly)) |>
  ungroup()

# Plot results to check assignment

VIC_map_MWR <- ggplot() +
   geom_sf(data = vicmap_sa1_G02_electorates_grouped_MWR,
        aes(geometry = geom, fill = Median_rent_weekly)) +
  theme_bw() +
  theme(legend.position = "bottom")
VIC_map_MWR

```
```{r Scoped view of Map MWR - Task 4}

# set the coordinate limits for the chosen electorates
crop_map_MWR <- vicmap_sa1_G02_electorates_grouped_MWR %>%
  st_crop(xmin = 144.8, xmax = 145.2,
        ymin = -38.1, ymax = -37.6)

cropped_map_MWR <- ggplot(crop_map) +
  geom_sf(aes(geometry = geom, fill = Median_rent_weekly))

cropped_map_MWR

```
```{r labelling the Map MWR - Task 4}

data_for_labels = crop_map_MWR %>%
     filter(DivisionNm %in% c("Kooyong", "Goldstein", "Melbourne", "Menzies","Maribyrnong", "Wills",
"Scullin", "McEwen", "Chrisholm"))

cropped_map_MWR +
  geom_sf_label(data = data_for_labels,
   aes(label = DivisionNm, geometry = geom),
   size = 2)

```
```{r Map data - Task 4}
aec_map <- read_sf(here::here("data/vic-july-2021-esri/E_VIC21_region.shp"))
```
```{r Wrangling data for Map M_tot_prsnl_inc_w - Task 4}

# Set projection to GDA1994 using EPSG:4283
st_crs(aec_map$geometry,4283)

# Transform projection from GDA1994 to GDA2020 using EPSG:7844
aec_map$geometry = st_transform(aec_map$geometry, 7844)

# Get SA1 Centroids
vicmap_sa1_G02_w_centroid <- vicmap_sa1_G02 |>
  mutate(centroid = st_centroid(geom))

# Determine with SA1 centroids overlap with which electoral regions
electoral_intersects = st_intersects(vicmap_sa1_G02_w_centroid$centroid,
                      aec_map$geometry,
                      sparse = FALSE)
arr_ind = which(electoral_intersects == TRUE, arr.ind = TRUE)
sa1_ind = arr_ind[,1]
division_ind = arr_ind[,2]
division_name = aec_map$Elect_div[division_ind]
sa1_name = vicmap_sa1_G02_w_centroid$SA1_NAME_2021[sa1_ind]

# Save the SA1 region and its matching electorate into a data frame
sa1_divisions = data.frame(SA1_NAME_2021 = sa1_name,
                    DivisionNm = division_name)

# Add the electorate names to the map
vicmap_sa1_G02_electorates  <- vicmap_sa1_G02 %>%
  right_join(sa1_divisions)

# Combine the data from electorates together using group_by
vicmap_sa1_G02_electorates_grouped_MTPIW = vicmap_sa1_G02_electorates |>
  group_by(DivisionNm) |>
  summarise(Median_tot_prsnl_inc_weekly = mean(Median_tot_prsnl_inc_weekly)) |>
  ungroup()

# Plot results to check assignment

VIC_map_MTPIW <- ggplot() +
   geom_sf(data = vicmap_sa1_G02_electorates_grouped_MTPIW,
        aes(geometry = geom, fill = Median_tot_prsnl_inc_weekly)) +
  theme_bw() +
  theme(legend.position = "bottom")
VIC_map_MTPIW

```
```{r Scoped view of Map MTPIW - Task 4}

# set the coordinate limits for the chosen electorates
crop_map_MTPIW <- vicmap_sa1_G02_electorates_grouped_MTPIW %>%
  st_crop(xmin = 144.8, xmax = 145.2,
        ymin = -38.1, ymax = -37.6)

cropped_map_MTPIW <- ggplot(crop_map_MTPIW) +
  geom_sf(aes(geometry = geom, fill = Median_tot_prsnl_inc_weekly))

cropped_map_MTPIW

```
```{r labelling the Map MTPIW - Task 4}

data_for_labels = crop_map_MTPIW %>%
     filter(DivisionNm %in% c("Kooyong", "Goldstein", "Melbourne", "Menzies","Maribyrnong", "Wills",
"Scullin", "McEwen", "Chrisholm", "Fraser", "Higgins", "Hotham"))

cropped_map_MTPIW +
  geom_sf_label(data = data_for_labels,
   aes(label = DivisionNm, geometry = geom),
   size = 2)

```
~~~~

ChatGPT

You can create a function to avoid repeating the code for each task. Here's a way to do it:

```r
library(sf)
library(dplyr)
library(ggplot2)

# Define a function to process map data
process_map_data <- function(map_data, property, coords, divisions) {
  # Set projection to GDA1994 using EPSG:4283
  st_crs(map_data$geometry, 4283)

  # Transform projection from GDA1994 to GDA2020 using EPSG:7844
  map_data$geometry = st_transform(map_data$geometry, 7844)

  # Get SA1 Centroids
  vicmap_sa1_G02_w_centroid <- vicmap_sa1_G02 %>%
    mutate(centroid = st_centroid(geom))

  # Determine which SA1 centroids overlap with which electoral regions
  electoral_intersects = st_intersects(vicmap_sa1_G02_w_centroid$centroid,
                                       map_data$geometry,
                                       sparse = FALSE)
  arr_ind = which(electoral_intersects == TRUE, arr.ind = TRUE)
  sa1_ind = arr_ind[,1]
  division_ind = arr_ind[,2]
  division_name = map_data[[divisions]][division_ind]
  sa1_name = vicmap_sa1_G02_w_centroid$SA1_NAME_2021[sa1_ind]

  # Save the SA1 region and its matching electorate into a data frame
  sa1_divisions = data.frame(SA1_NAME_2021 = sa1_name,
                             DivisionNm = division_name)

  # Add the electorate names to the map data
  vicmap_sa1_G02_electorates <- vicmap_sa1_G02 %>%
    right_join(sa1_divisions)

  # Combine the data from electorates together using group_by
  grouped_data <- vicmap_sa1_G02_electorates %>%
    group_by(DivisionNm) %>%
    summarise_at(vars({{property}}), ~mean(., na.rm = TRUE)) %>%
    ungroup()

  # Plot results to check assignment
  map_plot <- ggplot() +
    geom_sf(data = grouped_data,
            aes(geometry = geom, fill = {{property}})) +
    theme_bw() +
    theme(legend.position = "bottom")

  # set the coordinate limits for the chosen electorates
  cropped_map <- grouped_data %>%
    st_crop(xmin = coords[1], xmax = coords[2],
            ymin = coords[3], ymax = coords[4])

  cropped_plot <- ggplot(cropped_map) +
    geom_sf(aes(geometry = geom, fill = {{property}}))

  # Labelling the Map
  data_for_labels <- cropped_map %>%
    filter(DivisionNm %in% c("Kooyong", "Goldstein", "Melbourne", "Menzies", "Maribyr

  labelled_plot <- cropped_plot +
    geom_sf_label(data = data_for_labels,
                  aes(label = DivisionNm, geometry = geom),
                  size = 2)

  list(map_plot = map_plot, cropped_plot = cropped_plot, labelled_plot = labelled_plo
}

# Usage for Map MWR
map_MWR <- process_map_data(aec_map, Median_rent_weekly, c(144.8, 145.2, -38.1, -37.6

# Usage for Map MTPIW
map_MTPIW <- process_map_data(aec_map, Median_tot_prsnl_inc_weekly, c(144.8, 145.2, -
```

Replace `Median_rent_weekly` and `Median_tot_prsnl_inc_weekly` with the actual column names you are using. This function `process_map_data` will take the map data, the property you are interested in (like rent or personal income), the coordinates for cropping the map, and the column name for divisions. It will then generate the map, cropped map, and labelled map plots for you.

Function for Task 4