ANALYZING CANADIAN DEMOGRAPHIC AND HOUSING DATA

Building skills and community to analyze Canadian demographic and housing data

Jens von Bergmann

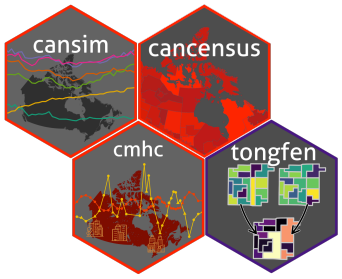


Table of contents

Preface **1**
Project based approach . . . 1
Goals . . . 2
Why use R? . . . 3
Building a Canadian data community . . . 4

1. Introduction **5**
1.1. A hypothetical example . . . 5
1.2. What you will learn in this book . . . 13

I. Getting started with R and RStudio **15**
R and RStudio . . . 16
Packages . . . 16
Basic data manipulation patterns . . . 16
Exploring the data . . . 17
Basic data manipulation . . . 18
Visualizing data . . . 19
More data manipulations . . . 21
Canadian data packages . . . 24

2. Introduction to the cansim package **25**

3. Introduction to the cancensus package **27**

4. Introduction to the cmhc package **30**

5. Introduction to the tongfen pacakge **32**

II. Basic descriptive analysis **34**

6. Geography of CERB **36**
6.1. Data sources . . . 36

6.2. Data acquisition . . . 36
6.3. Data preparation . . . 37
6.4. Analysis and visualization . . . 37
6.5. Interpretation . . . 41

7. Cars vs SUVs in Canada **42**
7.1. Question . . . 42
7.2. Data sources . . . 42
7.3. Data acquisition . . . 44
7.4. Data preparation . . . 45
7.5. Analysis and visualization . . . 45
7.6. Interpretation . . . 51

8. Under construction **52**
8.1. Data sources . . . 52
8.2. Data acquisition . . . 52
8.3. Data preparation . . . 53
8.4. Analysis and visualization . . . 53
8.5. Interpretation . . . 56

9. Geography of income change **57**
9.1. Data sources . . . 57
9.2. Data acquisition . . . 58
9.3. Data preparation . . . 59
9.4. Analysis and visualization . . . 60
9.5. Data acquisition (part 2) . . . 61
9.6. Analysis and visualization . . . 62
9.7. Interpretation . . . 63

III. Advanced descriptive analysis **64**

10. BC migration **66**
10.1. Question . . . 66
10.2. Data sources . . . 66
10.3. Data acquisition . . . 67
10.4. Data preparation . . . 68
10.5. Analysis and visualization . . . 69
10.6. Interpretation . . . 76

References **77**

Preface

This book is intended for people interested in learning how to access, process, analyze, and visualize Canadian demographic, economic, and housing data using R. The target audience we have in mind ranges from interested individuals interested in understanding their environment through data, community activists and community groups interested in introducing data-based approached into their work, journalists who want to report on data in their stories or aim to incorporate their own descriptive data analysis, non-profits or people involved in policy who are looking for data-based answers to their questions.

The most important prerequisite is a keen interest in using data to help understand how housing and demographics shape cities and rural areas in Canada, and a willingness to learn. Prior knowledge of R is not necessary, but may be beneficial.

Canada has high quality demographic, economic and housing data. While significant data gaps exist, the available data often remains under-utilized in policy and planning analyses. Moreover, many analyses that do come out go quickly out of date and can't easily be updated because they rely on non-reproducible and non-adaptable workflows.

In this book we will maintain a strong emphasis on reproducible and adaptable work flows to ensure the analysis is transparent, can easily be updated as new data becomes available, and can be tweaked or adapted to address related questions.

Project based approach

This book will take a project based approach to to teach through examples, with one project per section. Each project will be loosely broken up into four parts.

1. **Formulating the question.** What is the question we are interested in? Asking a clear question will help focus our efforts and ensure that we don't aimlessly trawl through data.
2. **Identifying possible data sources.** Here we try to identify data sources that can speak to our question. We will also take the time to read up on definitions and background concepts to better understand the data and prepare us for data analysis,

and understand how well the concepts in the data match our original question from step 1.

3. **Data acquisition.** In this step we will import the data into our current working session. This could be as simple as an API call, or more complicated like scraping we table from the web, or involve even more complex techniques to acquire the data.
4. **Data preparation.** In this step we will reshape and filter the data to prepare it for analysis.
5. **Analysis.** This step could be as simple as computing percentages or even doing nothing, is the quantities we are interested in already come with the dataset, if our question can be answered by a simple descriptive analysis. In other cases, when our question is more complex, this step may be much more involved. The book will try to slowly build up analysis skills along the way, with increasing complexity of questions and required analysis.
6. **Visualization.** The final step in the analysis process is to visualize and communicate the results. In some cases this can be done via a table or a couple of paragraphs of text explaining the results, but in most cases it is useful to produce graphs or maps or even interactive visualizations to effectively communicate the results.
7. **Interpretation.** What's left to wrap this up is to interpret the results. How does this answer our question, where does it fall short. What does this mean in the real-world context? What new questions emerge from this?

While we won't always follow this step by step process to the letter, it will be our guiding principle throughout the book. Sometimes things won't go so clean, where after the visualization step we notice that something looks off or is unexpected, and we may jump back up a couple of steps and add more data and redo parts of the analysis to better understand our data and how it speaks to our initial questions. We might even come to understand that our initial question was not helpful or was ill-posed, and we will come back to refine it.

Goals

By taking this approach we have several goals in mind:

- Stay motivated by using real world Canada-focused and (hopefully) interesting examples.
- Teach basic data literacy, appreciate definitions and quirks in the data.
- Expose the world of Canadian data and make it more accessible.

- Learn how data can be interpreted in different ways, and data and analysis is not necessarily "neutral".
- Learn how to effectively communicate results.
- Learn how to adapt and leverage off of previous work to answer new questions.
- Learn how to reproduce and critique data analysis.
- Build a community around Canadian data, where people interested in similar questions, or people using the same data, can learn from each other.
- Raise the level of understanding of Canadian data and data analysis so we are better equipped to tackle the problems Canada faces.

This is setting a very high goal for this book, and we are not sure we can achieve all of this. But we will try our best to be accessible and interesting as possible.

Why use R?

Most people reading this book will not have used R before, or only used it peripherally, maybe during a college course many years in the past. Instead, readers may be familiar with working through housing and demographic data in Excel or similar tools. Or making maps in QGIS or similar tools when dealing with spatial data. And the type of analysis outlined above that this book will teach can in general terms be accomplished using these tools.

But where tools like spreadsheets and desktop GIS fall short is in another important focus of this book: **transparency**, **reproducibility**, and **adaptability**.

An analysis in a spreadsheet or desktop GIS typically involves a lot of manual steps, the work is not **reproducible** without repeating these steps. We can't easily inspect how the result was derived, the analysis lacks **transparency**. When we just compute a ratio or percentage this may not be so bad, but trying to understand how a more complex analysis was done in a spreadsheet easily turns into a nightmare. Analysis that involves a lot of manual steps is not auditable without putting in the work to repeat those manual steps.

But why does this matter? It's always been this way, some experts produce analysis and produce a glossy paper to present the results. One can argue if this was an adequate modus operandi in the past, but we feel strongly that it's not in today's world. The lines between experts and non-experts has become blurred, and the value we place on lived experience has increased relative to more formal expertise. We argue this places different demands on policy-relevant analysis, it needs to be open and transparent, in principle anyone should be able to understand how the analysis was done and the conclusions were reached. That's

where reproducibility and transparency come in. And it also requires bringing up data analysis skills in the broader population, so that the ability to reproduce and critique an analysis in principle can be realized in practice.

The remaining reason for using R, **adaptability**, has also become increasingly important. The amount of data available to us has increased tremendously, but our collective ability to analyse data and extract information has not kept up. Doing analysis in R allows us to efficiently reuse previous analysis to perform a similar one. Or to build on previous analysis to deepen it. Which turbocharges our ability to do analysis, covering more ground and going deeper.

R is not the only framework to do this in, there are other options like python or julia. But we believe that R is best suited for people transitioning into this space, and we can rely on an existing ecosystem of packages to access and process Canadian data. People already proficient in python will have no problem translating what we do into their preferred framework, or dynamically switch back and forth between R, python or whatever other tools they prefer as needed and convenient.

Building a Canadian data community

Which brings us to our most ambitious goal, to help create a community around Canadian data analysis. When analysis is transparent, reproducible and adaptable people can piggyback of each other's work, reusing parts of analysis others have done and building and improving upon it. Or Critiquing and correcting analysis, or taking it toward a different direction. A community that grows in their understanding of data, and a community using a shared set of tools to access and process Canadian data, enabling discussions to move forward instead of in circles. A community that builds up expertise from the bottom up.

The book tries to address both of these requirements for building a Canadian data community, a principled approach to data and data analysis, while introducing R as a common framework to work in hoping that the reader will come away with

- better data literacy skills to understand and critique data analysis,
- technical skills to reproduce and perform their own data analysis, and
- a common tool set for acquiring, processing and analyzing Canadian data that facilitates collaborative practices.

1. Introduction

In this section we give a taste of what's to come. Some of the concepts introduced in the preface may be too abstract to picture for people just starting out in this space. People probably grasp the importance of having a principled approach to data analysis, from formulating a question all the way to sharing results. But why so much emphasis on reproducibility and adaptability? And do we really need to learn a new framework like R for this?

This is best understood by walking through a simple example of what analysis of Canadian data in R, and a Canadian data community might look like. We won't explain all steps in full detail here, this is to serve to illustrate the concepts talked above in the preface and give the reader a taste of what's to come.

If you don't understand all the code now, don't worry, that's part of the point of this book. We will work out and explain these examples in detail in the first chapter of the book. What's important right now is to illustrate the principle of reproducible and adaptable code, and how this can function to foster a community of Canadian data analysis. And to note how little code is needed to make this work.

1.1. A hypothetical example

Imagine Amy, a Toronto-based social services worker looking to pilot a community intervention targeted at children in low income. She is in the process of putting together a proposal describing her intervention and is trying to locate a good neighbourhood for her pilot and make a compelling case to possible funders.

Amy knows that census data has a good geographic breakdown of children in poverty, but the latest available data is from 2016, using 2015 income data. CRA tax data is available up to 2019, but also has information on families in low income, but nothing directly on children in the standard release tables at fine geographies. As a first step she settles on census data, with the goal to re-run the analysis once the 2021 data comes out later in the year.

She refers to the Census Dictionary to understand the various low income measures, and uses CensusMapper's interactive map that allows to explore these concepts. She would

have liked to use the Market Based Measure, but due to data availability she settles for LICO-AT.

She sets up a new Notebook and loads in the R libraries that she will need for this, ggplot2for graphing and **cancensus** for ingesting the data.

```
library(cancensus)
library(ggplot2)
```

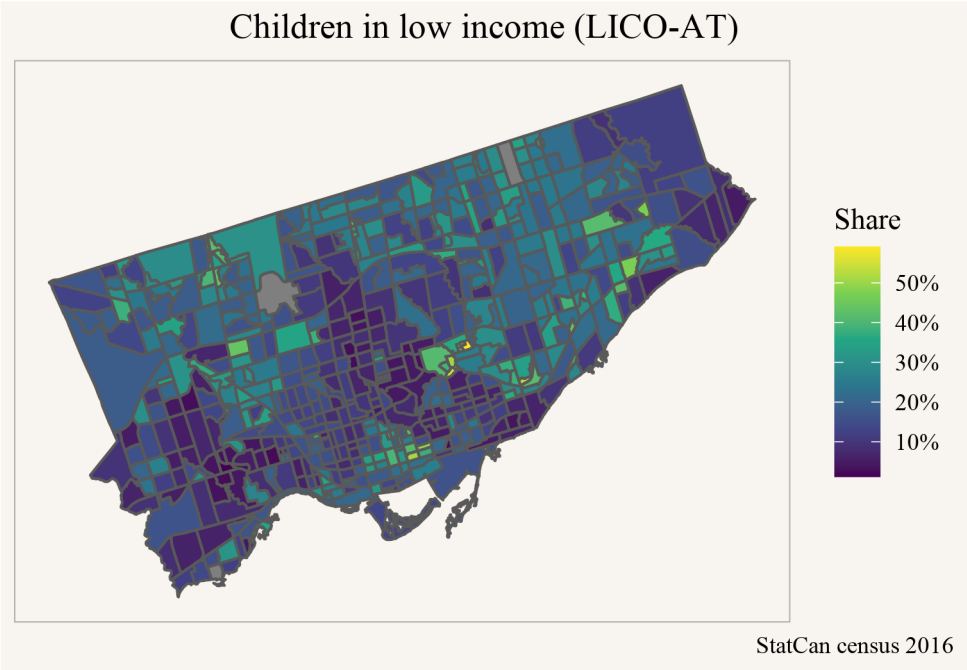
Next she pull in the data. the CensusMapper API GUI tool helps her locate the StatCan geographic identifier for Toronto, (3520005), and the internal CensusMapper vector for the percentage of children in LICO-AT (v_CA16_2573).

```
lico_yyz <- get_census("CA16",regions=list(CSD="3520005"), vectors=c(lico="v_CA16_2573"),
                       level="CT",geo_format="sf")
```

Here Amy specified that she wants data for the 2016 Canadian census ("CA16"), the region and vectors, at the census tract ("CT") level, with geographies as well as the low income data.

Now that she has the data at her finger times her first step is to make a map. For that she needs to tell **ggplot** is what variable to use as fill colour, and maybe give it a nicer colour scale and some labels to explain what the map is about.

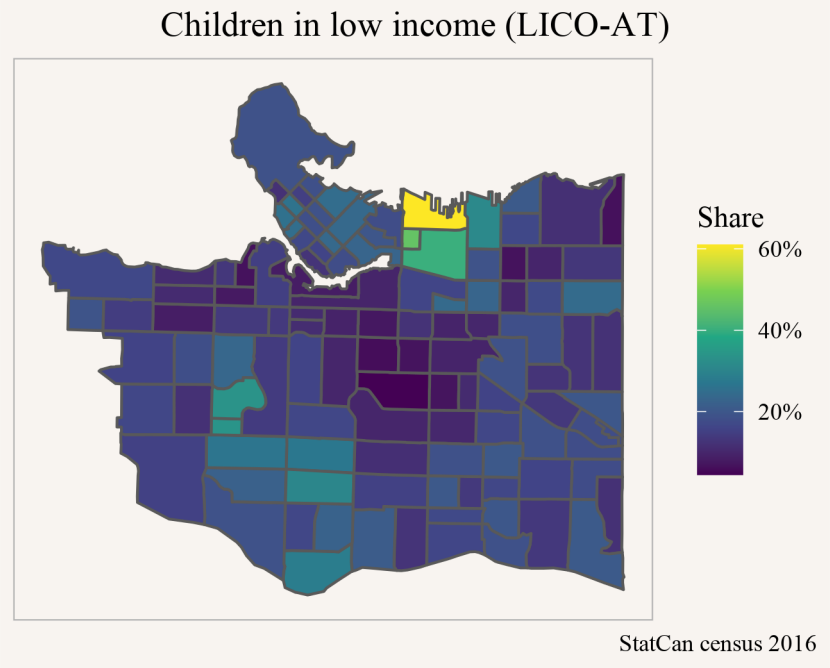
```
ggplot(lico_yyz, aes(fill=(lico/100))) +
  geom_sf() +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="Children in low income (LICO-AT)",
       fill="Share",
       caption="StatCan census 2016")
```



Based on this she locates a couple of good candidate neighbourhoods for her pilot and sends the map in a email to her colleague Peter to get input on which neighbourhood might be best suited.

Peter has some good feedback for Amy, but also gets an idea to try and set up something similar in Vancouver. Peter asks Amy if she can share the code, and Amy sends along the above code snippets. Peter looks up the geographic identifier for Vancouver and subs that in instead of Toronto's.

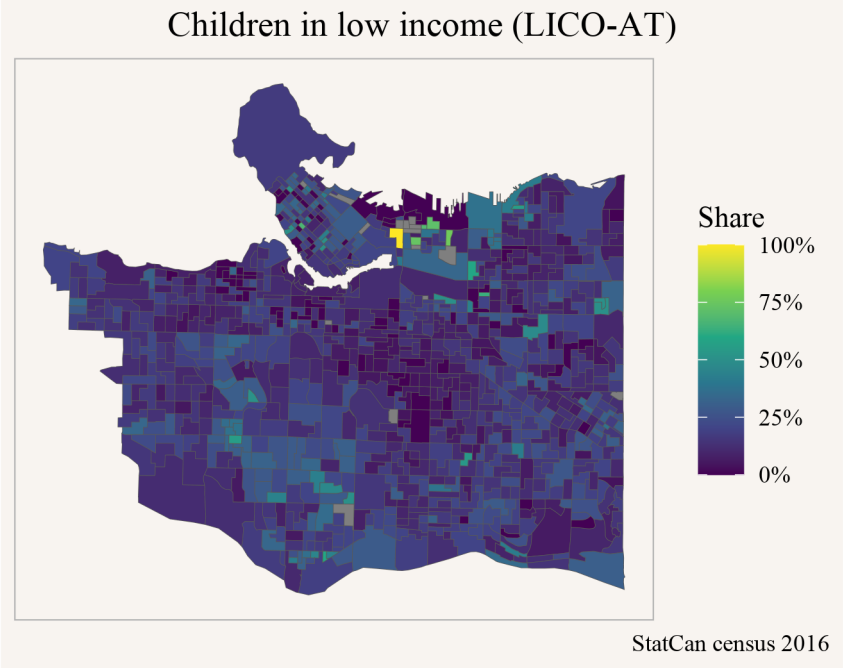
```
lico_yvr <- get_census("CA16",regions=list(CSD="5915022"), vectors=c(lico="v_CA16_2573"),
                       level="CT",geo_format="sf")

ggplot(lico_yvr, aes(fill=(lico/100))) +
  geom_sf() +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="Children in low income (LICO-AT)",
       fill="Share",
       caption="StatCan census 2016")
```



Easy peasy, thanks to Amy's previous work. Peter takes the map to his friend Yuko and asks her for advice where a community-based intervention for low-income children might make sense in Calgary. Yuko asks for the code from Peter to take a closer look herself.

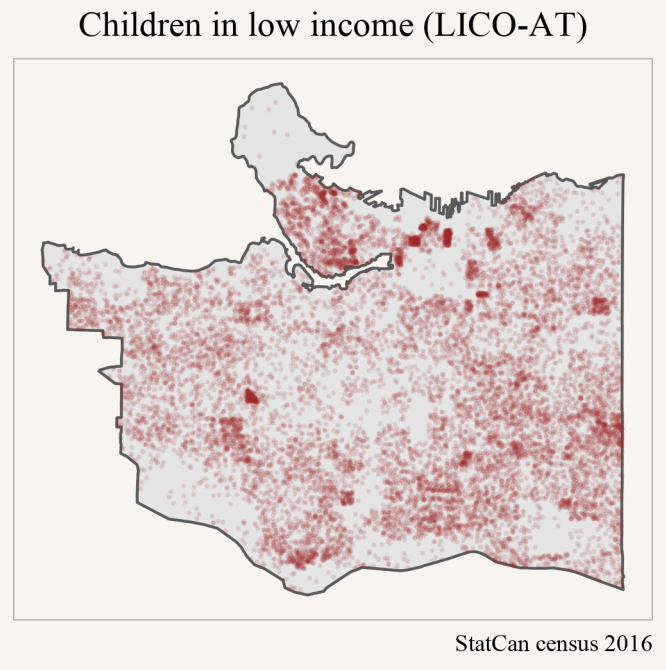
Yuko is interested in a finer geographic breakdown, so she swaps our the geographic level from census tracts to dissemination areas.

```
lico_yvr_da <- get_census("CA16",regions=list(CSD="5915022"), vectors=c(lico="v_CA16_2573")
                          level="DA",geo_format="sf")

ggplot(lico_yvr_da, aes(fill=(lico/100))) +
  geom_sf(size=0.1) +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="Children in low income (LICO-AT)",
       fill="Share",
       caption="StatCan census 2016")
```



But then Yuko pauses to think that maybe looking at share of the low income population is not the right metric. She decides to query the number of children in low income (vector "v_CA16_2558") and prepare the data for a dot-density map.

```
# remotes::install_github("mountainmath/dotdensity")
library(dotdensity)

lico_dots_yvr <- get_census("CA16",regions=list(CSD="5915022"),geo_format="sf",
                            vectors=c(lico="v_CA16_2558"), level="DA") |>
  compute_dots("lico")
yvr_city <- get_census("CA16",regions=list(CSD="5915022"),geo_format="sf")

ggplot(lico_dots_yvr) +
  geom_sf(data = yvr_city) +
  geom_sf(size=0.25,colour="brown",alpha=0.1) +
  coord_sf(datum=NA) +
  labs(title="Children in low income (LICO-AT)",
       fill="Share",
       caption="StatCan census 2016")
```



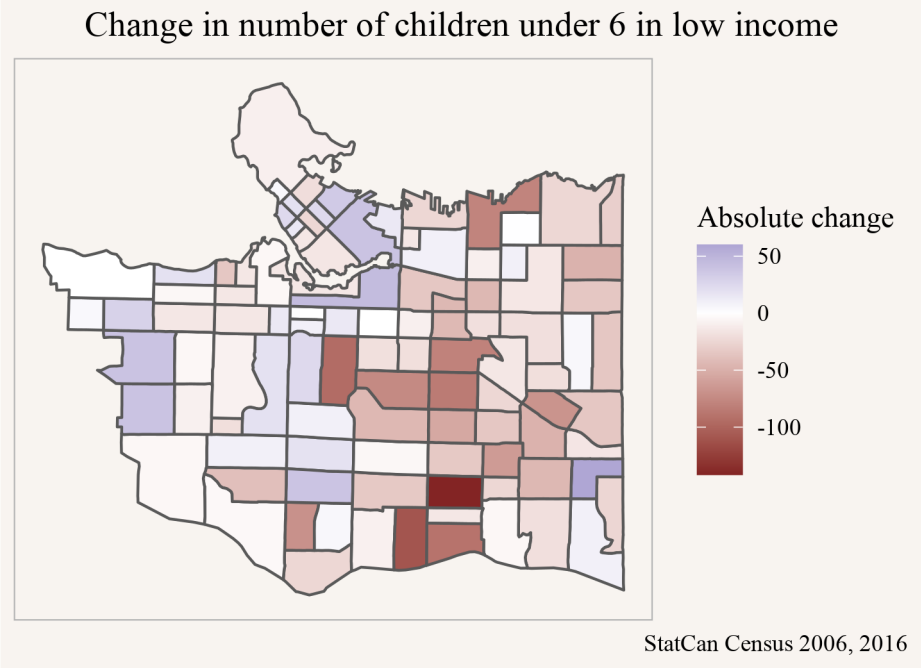
That paints a somewhat different picture, and Yuko feels this is much better suited to pinpoint where to best stage a community intervention. She lets Peter and Amy know and emails them her modifications to the code.

Meanwhile, Yuko's Vancouver friend Stephanie is looking specifically at children below the age of 6 in low income, and wants to understand how the geographic distribution of low income children has changed over time. Comparing census data through time can be tricky because census geographies change, but this problem has been completely solved via the **tongfen** R package. Looking at Yuko's work she thinks it might be best to look at both, the change in share of children in low income as well as the change in absolute number.

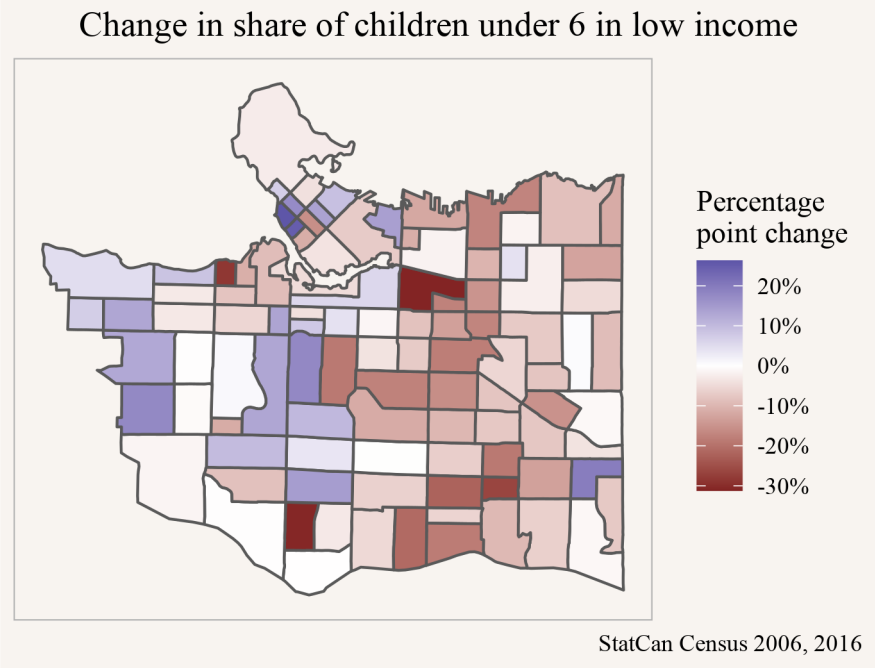
```
library(tongfen)
meta <- meta_for_ca_census_vectors(c(total_2006="v_CA06_1982",lico_share_2006="v_CA06_1984"
                                     lico_2016="v_CA16_2561",lico_share_2016="v_CA16_2576")

lico_data <- get_tongfen_ca_census(regions=list(CSD="5915022"),meta,level="CT") |>
  mutate(lico_2006=total_2006*lico_share_2006/100) |>
  mutate(`Absolute change`=lico_2016-lico_2006,
         `Percentage point change`=lico_share_2016-lico_share_2006)
```

Armed with this data Stephanie can plot the absolute and percentage point change in children below 6 in low income.

```
ggplot(lico_data,aes(fill=`Absolute change`)) +
  geom_sf() +
  scale_fill_gradient2() +
  coord_sf(datum=NA) +
  labs(title="Change in number of children under 6 in low income",
       caption="StatCan Census 2006, 2016")
```




```
ggplot(lico_data,aes(fill=`Percentage point change`/100)) +
  geom_sf() +
  scale_fill_gradient2(labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="Change in share of children under 6 in low income",
       fill="Percentage\npoint change",
       caption="StatCan Census 2006, 2016")
```

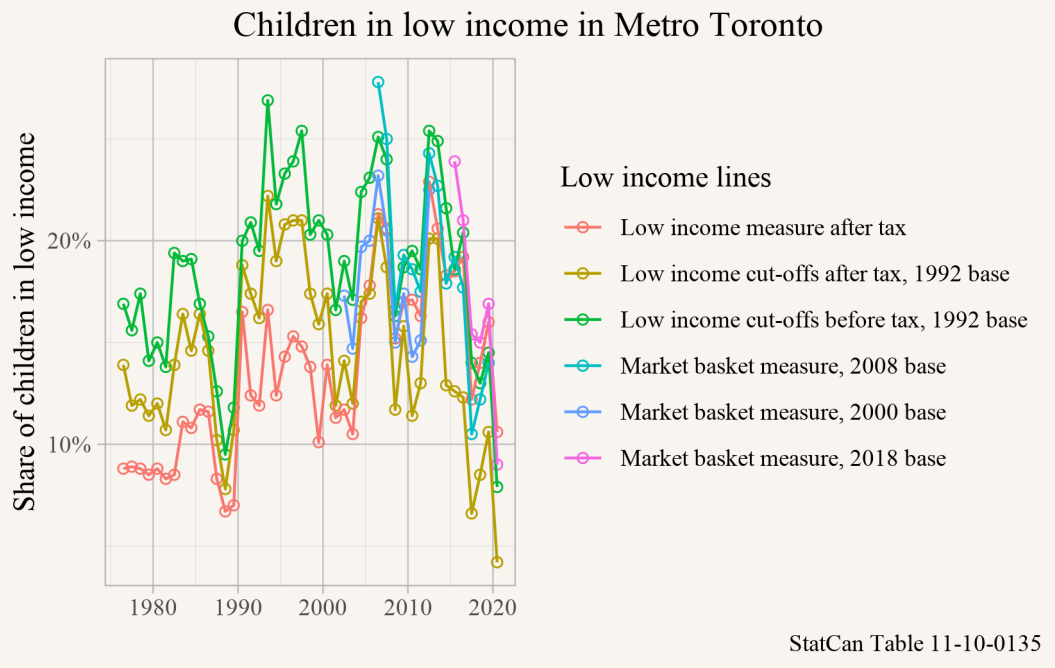


Stephanie shares her results with Amy in Toronto in case there are components of Amy's pilot specifically targeting children below 6 in low income.

Meanwhile Amy has been trying to understand more broadly how the share of low income children has evolved since the 2016 census (using 2015 income data) at the metropolitan level over longer time spans, so she looks through the StatCan socioeconomic tables and settles on table 11-10-0135, which also allows her to compare various low income concepts.

```
library(cansim)
mbm_timeline <- get_cansim("11-10-0135") |>
  filter(`Persons in low income`=="Persons under 18 years",
         GEO=="Toronto, Ontario",
         Statistics=="Percentage of persons in low income")

ggplot(mbm_timeline,aes(x=Date,y=val_norm,colour=`Low income lines`)) +
  geom_point(shape=21) +
  geom_line() +
  scale_y_continuous(labels=scales::percent) +
  labs(title="Children in low income in Metro Toronto",
       y="Share of children in low income",
```

```
        x=NULL,
        caption="StatCan Table 11-10-0135")
```



She notes that there has been a substantial overall drop in children in low income since 2015 across all measures, which is excellent news. She considers pushing off her pilot project until after the 2021 census data comes out to first understand if the geographic patterns have changed.

1.2. What you will learn in this book

Looking at R code for the first time can be intimidating. If the code looks opaque right now, there is no need to worry. It will be explained in detail in the first chapter and is very much part of the rationale for writing this book. If decisions around what low income metric to pick, or why **tongfen** is needed to compare census data through time are not clear, again, that will be explained in this book in detail and expanding understanding of data and data analysis is the other big rationale for this book.

Readers will learn how to reproduce analysis, how to critique analysis, and adapt it for their own purposes. And readers will learn how to conduct their own analysis in the Canadian context, based on questions and use cases relevant to them.

Hopefully the above hypothetical scenario have explained how the adaptability of the R code has made life much easier for several of the subsequent analysis steps, and how little code was needed to gain some insights and communicate results.

Part I.

Getting started with R and RStudio

Statistics Canada produces a lot of high quality demographic and economic data for Canada. CMHC complements this with housing data, and municipalities across Canada often provide relevant data through their Open Data portals.

R and RStudio

We will be working in R and the RStudio IDE, although using a different editor like Visual Studio Code works just as well, especially if you are already familiar with it. Within R we will be operating within the tidyverse framework, a group of R packages that work well together and allow for intuitive operations on data via pipes.

While an introduction to R is part of the goal of this book, an we will slowly build up skills as we go, we not give a systematic introduction but rather build up skills slowly as we work on concrete examples. It may be beneficial to supplement this with a more principled introduction to **R** and the **tidyverse.**

Packages

Packages are bundled sets of functionality that expand base R. We install or upgrade packages with the `install.packages`. For example, to install the tidyverse framework we type

```
install.packages("tidyverse")
```

into the R console. This will install or upgrade the package and required dependencies. To make the functionality, for example the `tibble` function from the **tibble** package that is part of **tidyverse**, available to use we can then either access functions from the package using the `::` namespace selector `tibble::tibble()` or first load the **tibble** or **tidyverse** package via `library(tidyverse)` that makes the `tibble()` function available without having to use the namespace selector.

Basic data manipulation patterns

There are several basic data manipulation patterns that we will use throughout, and we want to give a quick overview using the Palmer Penguins dataset from the **palmerpenguins** package.

```
# install.packages("palmerpenguins") # install the package if needed
library(palmerpenguins)
```

Now we have all the functionality of the **palmerpenguins** package available.

Exploring the data

With the **palmerpenguins** package comes the `penguins` dataset, we can expect the first few rows using the `head()` function which displays the first few rows.

```
head(penguins)
```

```
# A tibble: 6 x 8
  species island    bill_length_mm bill_depth_mm flipper_l~1 body_~2 sex    year
  <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
1 Adelie  Torgersen           39.1          18.7         181    3750 male   2007
2 Adelie  Torgersen           39.5          17.4         186    3800 fema~  2007
3 Adelie  Torgersen           40.3          18           195    3250 fema~  2007
4 Adelie  Torgersen           NA            NA            NA      NA <NA>   2007
5 Adelie  Torgersen           36.7          19.3         193    3450 fema~  2007
6 Adelie  Torgersen           39.3          20.6         190    3650 male   2007
# ... with abbreviated variable names 1: flipper_length_mm, 2: body_mass_g
```

The `str()` function offers another convenient way to get an overview over the data.

```
str(penguins)
```

```
tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
 $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
 $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
 $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
 $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
 $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

We can also type `View(penguins)` into the console to view the dataset in a spreadsheet form.

Basic data manipulation

To manipulate and visualize the data we load the tidyverse package.

```
library(tidyverse)
```

We will explore some common data manipulation and visualization workflows.

Count groups

To see how many rows there are for each species we 'pipe' the `penguins` dataset into the `count()` verb. Pipes are how we can stepwise transform data, the pipe operator is given by `%>%` withing the **tidyverse** framework and now also available natively in base R via `|>`. These two function (almost) the same way, and we will use both in this book.

```
penguins |> count(species)
```

```
# A tibble: 3 x 2
  species       n
  <fct>     <int>
1 Adelie      152
2 Chinstrap    68
3 Gentoo      124
```

This gives us the count of each species in the dataset, the pipe `|>` inserts the left hand side as the first argument in the `count()` function. We could have equivalently written this without the pipe operator as `count(penguins,species)`.

Group and summarize

The usefulness of the pipe operator becomes clear when we chain several data transformation. If we want to know the mean bill length by species, we group by species and summarize the data.

```
penguins |>
  group_by(species) |>
  summarize(bill_length_mm=mean(bill_length_mm, na.rm=TRUE))
```

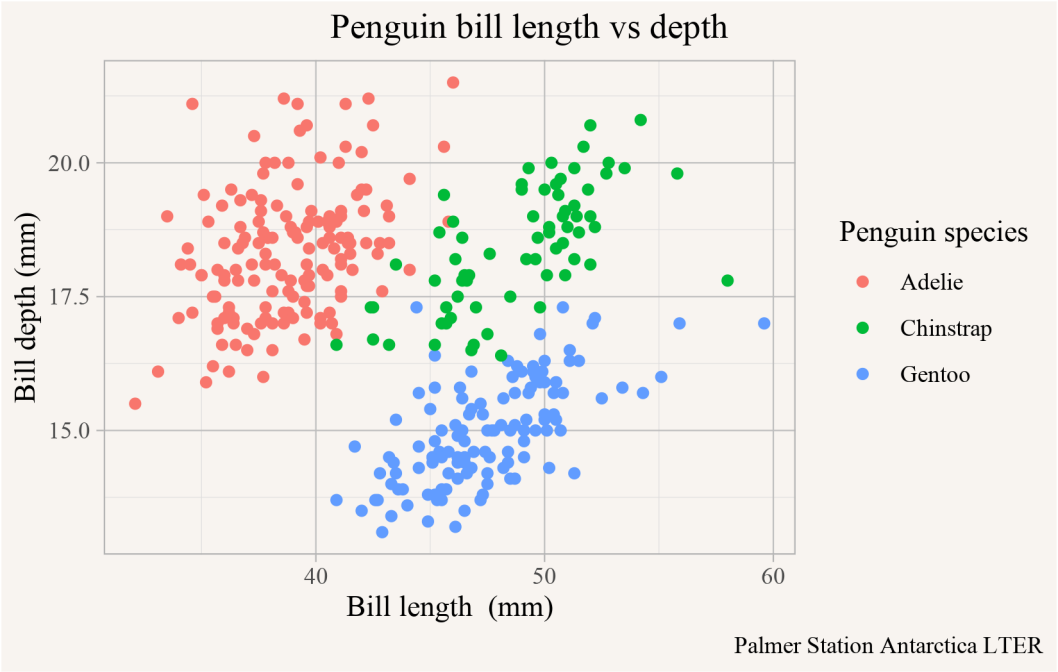
```
# A tibble: 3 x 2
  species   bill_length_mm
  <fct>              <dbl>
1 Adelie              38.8
2 Chinstrap           48.8
3 Gentoo              47.5
```

Here we explicitly specify how missing values should be treated when summarizing, `na.rm=TRUE` says that NA values should be ignored when computing the mean.

Visualizing data

We can visualize the data using `ggplot`. For this we have to specify the mapping aesthetics, we plot the bill length on the x-axis, the depth on the y-axis, colour by species and plot the data as points. The `labs()` function allows us to customize the graph labels.

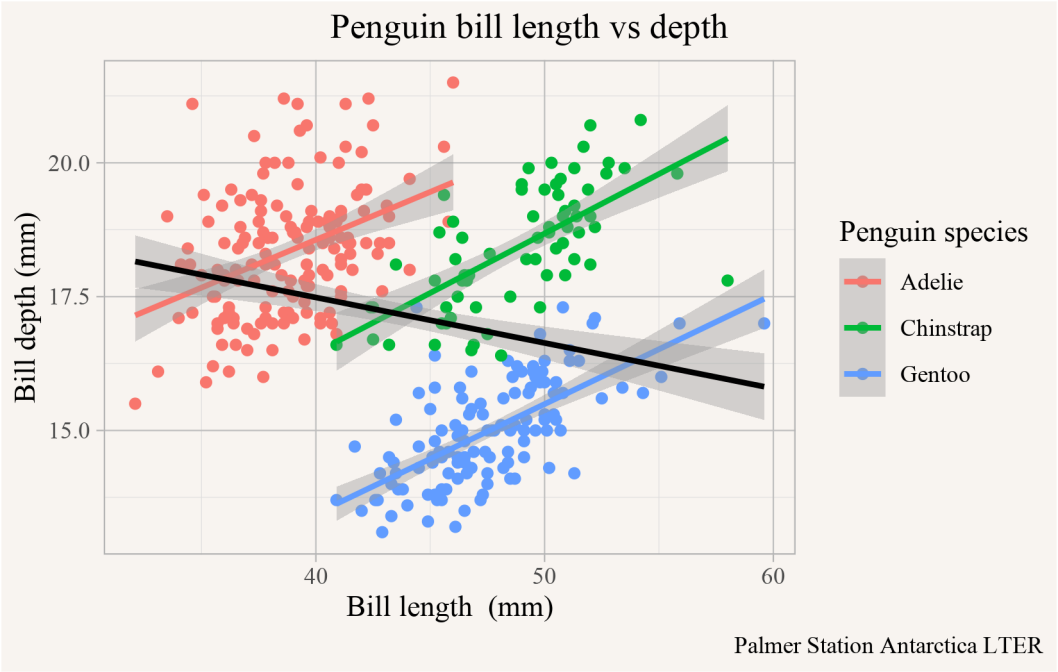
```
ggplot(penguins,aes(x=bill_length_mm,y=bill_depth_mm,colour=species)) +
  geom_point() +
  labs(title="Penguin bill length vs depth",
       x="Bill length  (mm)",y="Bill depth (mm)",
       colour="Penguin species",
       caption="Palmer Station Antarctica LTER")
```



Add regression lines

As an aside we note the Simpson's paradox, in the overall dataset the bill depth declines with length, but if we look separately within each species the bill depth increases with bill length. To make that explicit we can add regression lines using the `geom_smooth` function using `lm` (linear model) as the smoothing method.

```
ggplot(penguins,aes(x=bill_length_mm,y=bill_depth_mm,colour=species)) +
  geom_point() +
  geom_smooth(method="lm") +
  geom_smooth(method="lm", colour="black") +
  labs(title="Penguin bill length vs depth",
       x="Bill length  (mm)",y="Bill depth (mm)",
       colour="Penguin species",
       caption="Palmer Station Antarctica LTER")
```



The first `geom_smooth()` function will add a regression line for each species, distinguished by colour in the plot aesthetics. Overriding the `colour` argument in the second `geom_smooth()` function will forget that the data was coloured by species and add the black regression line run on the entire dataset.

More data manipulations

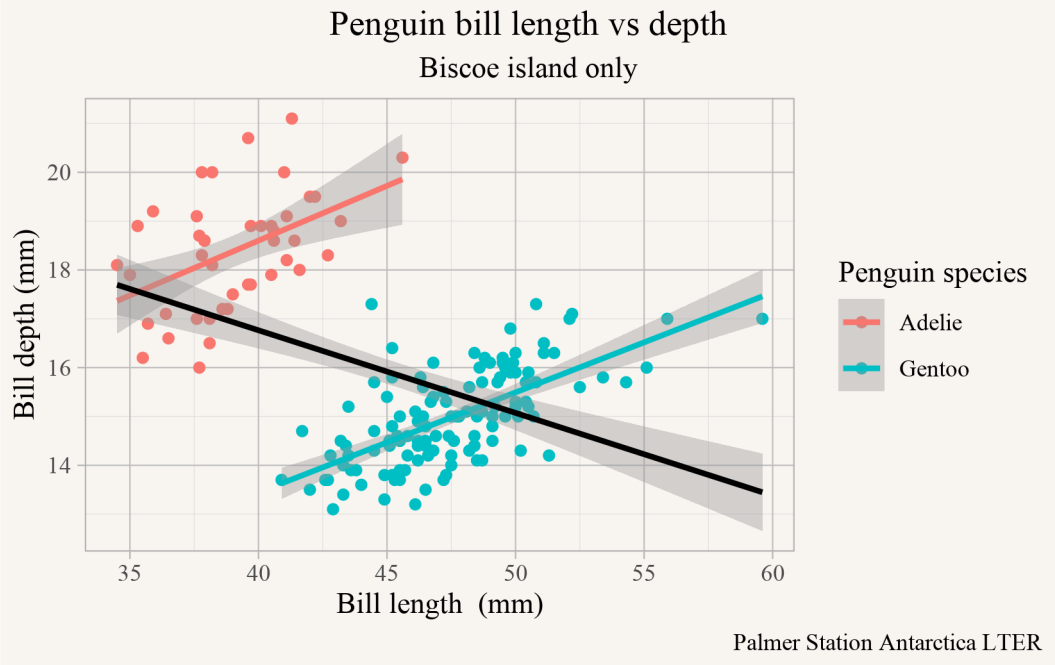
There are several common data manipulation steps that we will employ frequently.

Filtering rows

Often we are only interested in subsets of the data, we can filter the rows in the dataset by using the `filter` verb from the **dplyr** package that is part of **tidyverse**. For example, if we want to take the previous plot but only show it for penguins on the island of Biscoe we can filter the data accordingly before plotting it.

```
penguins |>
  filter(island=="Biscoe") |>
ggplot(aes(x=bill_length_mm,y=bill_depth_mm,colour=species)) +
```

```
  geom_point() +
  geom_smooth(method="lm") +
  geom_smooth(method="lm", colour="black") +
  labs(title="Penguin bill length vs depth",
       subtitle="Biscoe island only",
       x="Bill length  (mm)",y="Bill depth (mm)",
       colour="Penguin species",
       caption="Palmer Station Antarctica LTER")
```



Selecting columns

Instead of filtering rows It can be useful to select a subset of the columns to remove columns we don't need and de-clutter the dataset. This is especially useful when producing tables. If we want a table of the numeric data fields of all female Adelie penguins on the island of Biscoe observed in 2007 we can filter by sex and island and select the columns we want.

```
penguins |>
  filter(island=="Biscoe", sex=="female", species=="Adelie", year==2007) |>
  select(where(is.numeric),-year)
```

```
# A tibble: 5 x 4
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
           <dbl>         <dbl>             <int>       <int>
1           37.8          18.3               174        3400
2           35.9          19.2               189        3800
3           35.3          18.9               187        3800
4           40.5          17.9               187        3200
5           37.9          18.6               172        3150
```

Mutating data

We often want to change data fields, or compute new columns from existing ones. For example, if we want to convert the body mass from *g* to *kg* we can add a new column using `mutate` for that.

```
penguin_selection <- penguins |>
  filter(island=="Biscoe", sex=="female", species=="Adelie", year==2007) |>
  mutate(body_mass_kg=body_mass_g/1000) |>
  select(where(is.numeric),-year,-body_mass_g)

penguin_selection
```

```
# A tibble: 5 x 4
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_kg
           <dbl>         <dbl>             <int>        <dbl>
1           37.8          18.3               174         3.4
2           35.9          19.2               189         3.8
3           35.3          18.9               187         3.8
4           40.5          17.9               187         3.2
5           37.9          18.6               172         3.15
```

Pivoting data

The data in our `penguin_selection` dataset above is in **wide form**, all the different variables are in their own column. Often it is useful to convert it to **long form**, where we only have one value column with the numeric values and another column specifying the type of measurement. In this case it is useful to add an identification column so that we know which measurements belong to the same penguin. We can just label the penguins by row number.

```
penguin_selection_long <- penguin_selection |>
  mutate(ID=row_number()) |>
  pivot_longer(-ID,names_to="Metric",values_to="Value")

penguin_selection_long |> head()
```

```
# A tibble: 6 x 3
     ID Metric            Value
  <int> <chr>             <dbl>
1     1 bill_length_mm     37.8
2     1 bill_depth_mm      18.3
3     1 flipper_length_mm 174
4     1 body_mass_kg        3.4
5     2 bill_length_mm     35.9
6     2 bill_depth_mm      19.2
```

We can do the reverse transformation, going **from long** form to **wide form**, using `pivot_wider`.

```
penguin_selection_long |>
  pivot_wider(names_from = Metric,values_from = Value)
```

```
# A tibble: 5 x 5
     ID bill_length_mm bill_depth_mm flipper_length_mm body_mass_kg
  <int>          <dbl>         <dbl>              <dbl>        <dbl>
1     1           37.8          18.3                174         3.4
2     2           35.9          19.2                189         3.8
3     3           35.3          18.9                187         3.8
4     4           40.5          17.9                187         3.2
5     5           37.9          18.6                172         3.15
```

This recovers the previous form of the data, with the added `ID` column.

Canadian data packages

During the course of this book we will make heavy use of several R packages to facilitate data access to Canadian data, we will introduce them in this chapter.

2. Introduction to the cansim package

The **cansim** R package interfaces with the StatCan NDM that replaces the former CANSIM tables. It can be queried for

- whole tables
- specific vectors
- data discovery searching through tables

It encodes the metadata and allows to work with the internal hierarchical structure of the fields.

Larger tables can also be imported into a local SQLite database for reuse across sessions without the need to re-download the data, and better performance when subsetting the data or performing other basic data operations at the database level before loading the data into memory.

Data discovery is can be cumbersome, the `list_cansim_cubes` function from the **cansim** package fetches the newest list of all available tables and can be filtered by survey, release

date or dates of data coverage. The table list is cached for the duration of the R session. The `seach_cansim_cubes` function provides a convenient shortcut to narrow down this list.

In some cases searching the web for "StatCan Table xxxx", where "xxxx" contains search phrases for the data of interest, is sometimes a useful way to discover data. In reverse, we can bring up the StatCan webpage for a specific table number using the `view_cansim_webpage` function and explore the data via the web interface. Especially for large datasets this can be a faster way to determine if a specific table contains the information we are interested in without first having to download the data.

To get overview information for a table we have already downloaded the `get_cansim_table_overview` function provides a high-level overview over the variables contained in the table. The `get_cansim_column_list` function returns a list of the available columns or dimensions in the table, and `get_cansim_column_categories` returns the list of levels in a specific dimension. The `get_cansim_table_nots` provides the data notes that can hold important information to guide interpretation of some of the dimensions or levels.

The data is accessed via `get_cansim` function, or alternatively the `get_cansim_sqlite` function that stores the data permanently for use across R sessions in a local SQLite database. By default the English language tables are accessed, setting the `language="fr"` parameter changes that to the French version. The SQLite option is especially useful for larger tables. The **cansim** package will emit a warning if an SQLite table is outdated and newer data is available, if the `auto_refresh=TRUE` option is passed to the function call it will automatically download any new data if available. When accessing data from the SQLite version we can use normal `dplyr` verbs to filter the data or perform basic `select`, `group_by` or `summarize` operations before calling `collect_and_normalize` to fetch the result from the database and enrich it with metadata.

Metadata added by the **cansim** package includes converting the dimension values to factors and adding information on the hierarchical structure of the levels. Moreover, the package creates a native `Date` field and a `val_norm` field with normalized values. The values shipped by StatCan are sometimes expressed in "thousands of units", the `val_norm` converts this to base units for easier interpretation and uniformity across tables.

More information can be found in the package documentation and the package vignettes.

To install the package from cran use

```
install.packages("cansim")
```

3. Introduction to the cancensus package

The **cancensus** R package interfaces with the CensusMapper API server. It can be queried for

- census geographies for census years 1996 through 2021
- census profile data for census years 1996 through 2021
- some census custom tabulations
- hierarchical metadata of census variables
- some non-census data that comes on census geographies, e.g. T1FF taxfiler data

A slight complication, the **cancensus** packageneeds an API key. You can sign up for one for free on CensusMapper.

Once you have your API key it's useful to store it as an environment variable in your .Renviron configuration file so it's available in all your R sessions.

```
install.packages("cancensus")

cancensus::set_api_key(key = "CensusMapper_XXXX...XXXX", install=TRUE)
```

By default {cancensus} cached downloaded census data, which makes it easier and faster to re-run analysis and protects from overusing the CensusMapper API quota. To use caching an local path needs to be designated for data caching. The cache is shared across R sessions.

```
cancensus::set_cache_path(cache_path = "<local path ot cache data>", install=TRUE)
```

{cancensus} provides convenient access to census data. Calls to {cancensus} require to spectify

- The `dataset`, for example "CA21" for the 2021 Canadian census. A list of available datasets can be accessed via `cancensus::list_census_datasets()`.
- The `regions` to access the data for, this is a list keyed by geographic levels. For example, to access data for the Vancouver census metropolitan area it would be `list(CMA="59933")`, for the City of Toronto it would be `list(CSD="3520005")`. Region parameters can contain several regions of the same type or mix regions of different type. For example, to access data for the region covered by the Vancouver School Board, we need to assemble two CSDs and three CTs `list(CSD=c("5915022","5915803"),CT=c("9330069.04","9330069.03","9330069.02"))`. This allows pinpointing what geographic region we are interested in.
- The geographic `level` to query the data for. This simply be the regions specified in the `regions` parameter, but it could also be any geographic level equal to or lower than the lowest level geographic region specified in the `regions` parameter. Valid level identifiers are `DB` for dissemination blocks, `DA` for dissemination areas, `EA` for enumeration areas for the 1996 census, `CT` for census tracts, `CSD` for census subdivisions, `CMA` for census metropolitan areas or census agglormerations, `CD` for census districts, `PR` for provinces or territories and `C` for country level data. Geographic regions can also be assembled using the CensusMapper API GUI tool, CSD and higher level geographies can be explored or searched programmatically via the `list_census_regions()` or `search_census_regions()` functions.
- The `vectors` parameter allows to specify which census variables to query. By default the data comes with population, dwelling and household counts, other census variables can be explored and selected via the CensusMapper API GUI tool or explored

or searched programmatically via the `list_census_vectors()` or `find_census_vectors()` functions. There are also helper function to select variables using the internal CensusMapper metadata and hierarchy of census variables via the `child_census_vectors()` function. For finer control over the names of the names of the returned census variable the `vectors` parameter can be a named vector.

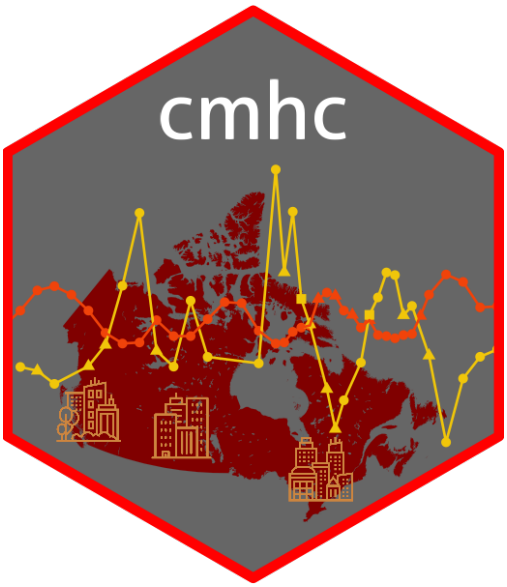
- The `geo_format` parameters allows to select if geographic data should also be downloaded, and if yes, in what format. In this post we will only access data via the modern "sf" format, but data can also be accessed in the legacy "sp" spatial data format.

As an example we will retried the share of the population in Toronto, Mississauga, and Brampton spending 30% or more of income on shelter costs in 2016.

```
library(cancensus)
library(dplyr)
regions <- list(CSD=c("3520005","3521005","3521010"))
vectors <- c(shelter_cost_burdened="v_CA16_4889", shelter_base = "v_CA16_4886")

data <- get_census(dataset = "CA16", regions=regions, vectors=vectors)
data %>%
  mutate(`Share shelter cost burdened`=shelter_cost_burdened/shelter_base) |>
  select(GeoUID,`Region Name`,`Share shelter cost burdened`)
```

```
  GeoUID  `Region Name`    `Share shelter cost burdened`
  <chr>   <fct>                                   <dbl>
1 3520005 Toronto (C)                             0.296
2 3521005 Mississauga (CY)                        0.264
3 3521010 Brampton (CY)                           0.305
```

4. Introduction to the cmhc package



The cmhc R package interfaces with the CMHC Housing Market Information Portal (HMIP) and allows programmatic and reproducible access to CMHC data. This gives access to data from four major CMHC surveys

- Starts and Completions Survey (Scss), which has data on housing construction covering starts, completions, units under construction, length of construction, absorbed and unabsorbed units and their prices.
- Rental Market Survey (Rms), which surveys the purpose-built rental market on an annual (and for some time twice-annual) basis. It has data on vacancy rates, availability rates, rents, fixed sample rent change and the overall rental universe by bedroom type, structure size, and year of construction.
- Secondary Rental Market Survey (Srms), which covers parts of the secondary market rental market with data on condominium apartment vacancy ratesm rents, and number and share of rented units, as well as some information on other secondary rentals.

- Senior's housing (Seniors), which gives data on seniors housing of various levels of assistance.
- Census data (Census), which holds several housing related cross-tabulations.

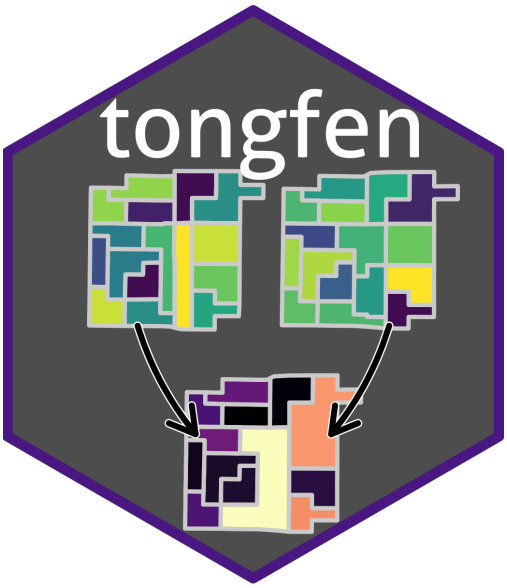
The package is designed to work in conjunction with the **cancensus** package and census geographic identifiers.

To install the package from CRAN use.

```
install.packages("cmhc")
```

The nature of the CMHC backend makes it at times challenging to find data, the **cmhc** package has several convenience functions to facilitate data discovery. The `list_…` functions, for example `list_cmhc_surveys()` list options. The `select_cmhc_table()` allows the interactive selection of data tables in the console, and returns the syntax for the desired function call to acquire the data.

5. Introduction to the tongfen pacakge



The tongfen R package facilitates making data on different geometries comparable.

TongFen ( ) means to convert two fractions to the least common denominator, typically in preparation for further manipulation like addition or subtraction. In English, that's a mouthful and sounds complicated. But in Chinese there is a word for this, TongFen, which makes this process appear very simple.

When working with geospatial datasets we often want to compare data that is given on different regions. For example census data and election data. Or data from two different censuses. To properly compare this data we first need to convert it to a common geography. The process to do this is quite analogous to the process of TongFen for fractions, so we appropriate this term to give it a simple name. Using the **tongfen** package, preparing data on disparate geographies for comparison by converting them to a common geography is as easy as typing `tongfen`.

In particular, the package has a number of convenience functions to facilitate making Canadian census data comparable through time, making it easy to perform longitudinal

analysis on fine geographies based on the Canadian Census. Essentially, the **tongfen** package creates a semi-custom tabulation based on Dissemination Block, Dissemination Area, or Census Tract geographies.

These semi-custom tabulations are created in three steps:

1. Create a correspondence table for geographies from different censuses. By default the official StatCan correspondence files are used for that, but these only exist back to 2001 when the current geographic system based on *dissemination blocks* and *dissemination areas* started. To go back further, when *enumeration areas* were the basic building block, we need to rely on geospatial matching of the areas to create a harmonized common geography.
2. Create metadata that contains information about how the census variables of interest can be aggregated in the case where geographies get joined. For example, if we are interested in the share of households in low income, we need to know what this share is based on in order to correctly aggregate it up. CensusMapper holds detailed metadata, so this process is automated.
3. Join geographies and aggregate census data as described in the correspondence table from Step 1 and the metadata in step 2.

The result of this process is a semi-custom tabulation of the data we want that is created on the fly, at the price of coming on a slightly coarser geography than the original input geographies in cases where geographies had to be joined to create the harmonized geography.

To install the package from CRAN use

```
install.packages("tongfen")
```

Part II.

Basic descriptive analysis

In this section we will look at how to do basic descriptive analysis. The questions we ask here will be quite simple, for example: How has income changed over time? Or: Which areas of Toronto have the highest incomes?

The accompanying analysis won't be very involved, sometimes we will compute percentages or make other simple data manipulations, but generally the analysis will be quite straightforward. We will focus on how to find data sources that can inform on our question, how to get the data, and how to present and interpret it.

6. Geography of CERB

In 2020 Canada introduced the COVID-19 Emergency Recovery Benefit (CERB), a program to support people with who lost income during the pandemic.

Where did CERB benefits go?

6.1. Data sources

Standard T1FF taxfiler data has this for large geographies, to understand fine geographic distribution we turn to Census data from the 2021 census, which reports on 2020 income. The census dictionary explains

> Canada Emergency Response Benefit (CERB) payments received during the reference period. This benefit was intended to provide financial support to employees and self-employed Canadians who had lost their job or were working fewer hours due to the COVID-19 pandemic and the public health measures implemented to minimize the spread of the virus.

Census income data is taken directly from T1 tax returns and linked at the individual person level.

6.2. Data acquisition

We can use the CensusMapper API tool and search for "COVID-19" in the *Variable Selection* tab to locate available census variables. Since we are interested in where people lived that received the benefit we select **v_CA21_593**, the number of recipients, as well as **v_CA21_554**, the baseline of people 15 years or older who are in principle eligible for this benefit.

We also need to decide which region we want to investigate, let's take a look at the City of Ottawa. We can select the city in the *Region Selection* tab and read off the geographic identifier **3506008** for the City of Ottawa in the *Overview* tab.

Now we have all we need to pull in the data, we just need to decide on the geographic granularity. Let's use **census tracts**, a standard geographic region aiming to capture between 2,500 and 7,500 people in metropolitan areas. We also specify that we want the geographies, not just the tabular data.

```
library(cancensus)
ottawa_cerb <- get_census("CA21",regions=list(CSD="3506008"),
                          vectors=c(cerb="v_CA21_593", base="v_CA21_554"),
                          level="CT", geo_format="sf")
```

6.3. Data preparation

To understand the geographic distribution we compute the percentage of people 15 years and over receiving CERB. Generally in this book we work in the **tidyverse** to help with data manipulation and visualization, so we load that library too.

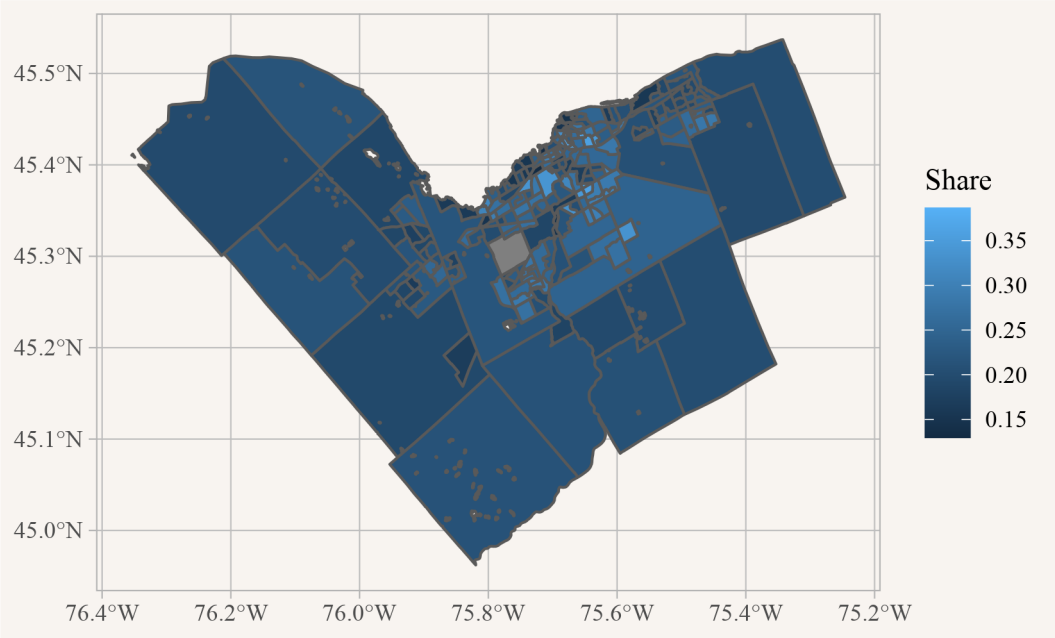
```
library(tidyverse)
plot_data <- ottawa_cerb |>
  mutate(Share=cerb/base)
```

There is not much to do, computing a percentage is a simple division. The `mutate` verb creates a new column called `Share` holding the computed ratios.

6.4. Analysis and visualization

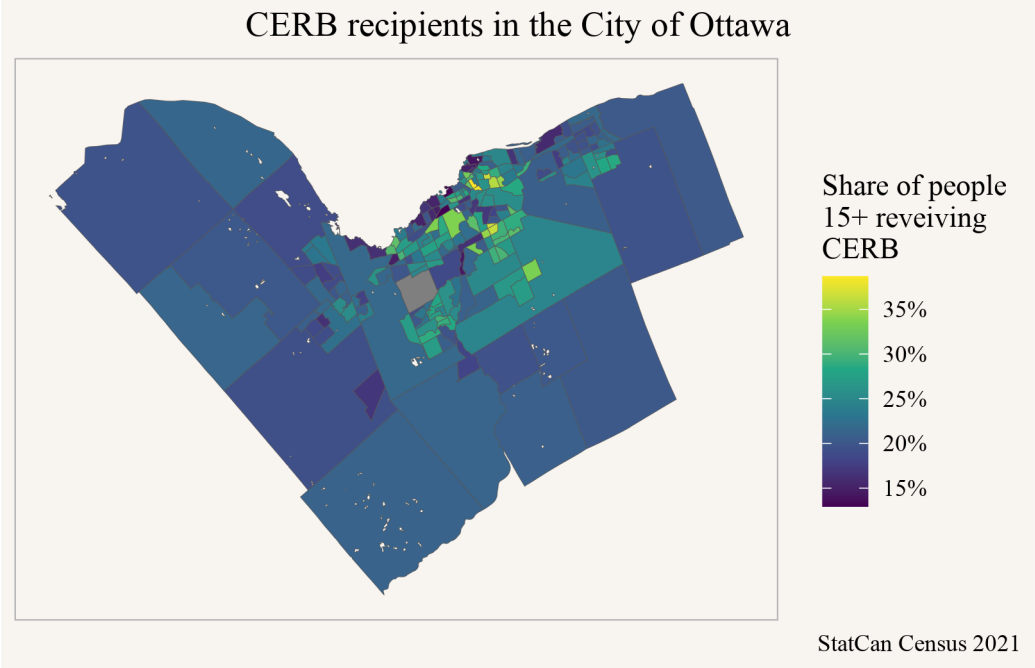
All that's left is to visualize the data. To plot geographic data we use `ggplot` and the `geom_sf` geometry. We need to tell it how to colour the map, the *aesthetic*, and we specify to fill each area by the share of CERB recipients.

```
ggplot(plot_data) +
  geom_sf(aes(fill=Share))
```



To make this a little nicer we add labels, remove the coordinate grid and choose nicer colours and reduce the boundary line size.

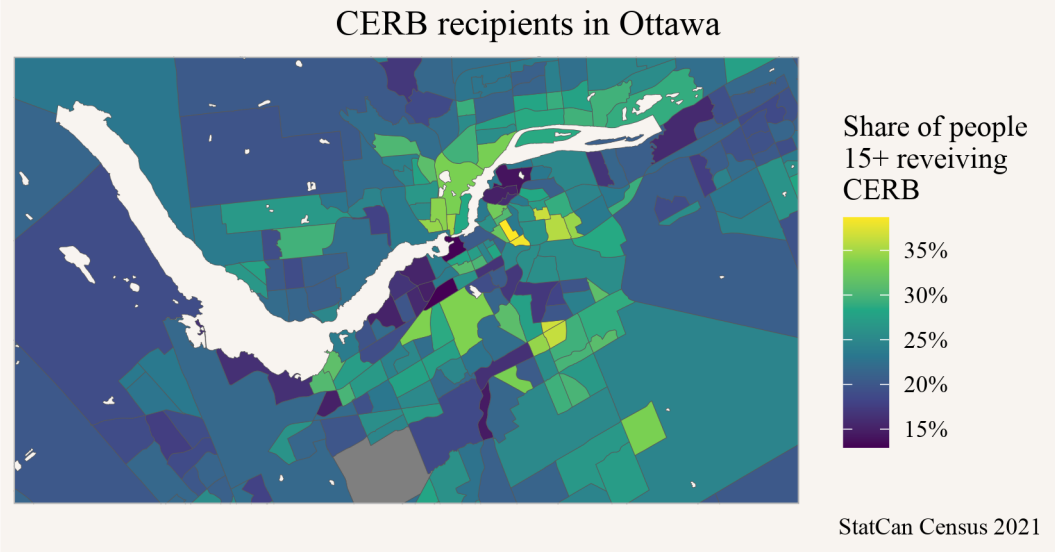
```
ggplot(plot_data) +
  geom_sf(aes(fill=Share), size=0.1) +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="CERB recipients in the City of Ottawa",
       fill="Share of people\n15+ reveiving\nCERB",
       caption="StatCan Census 2021")
```



It is difficult to see the central parts, we might want to zoom in a little. At the same time, it might be useful to add in Gatineau and surrounding municipalities, so maybe we want the data for the entire metro area.

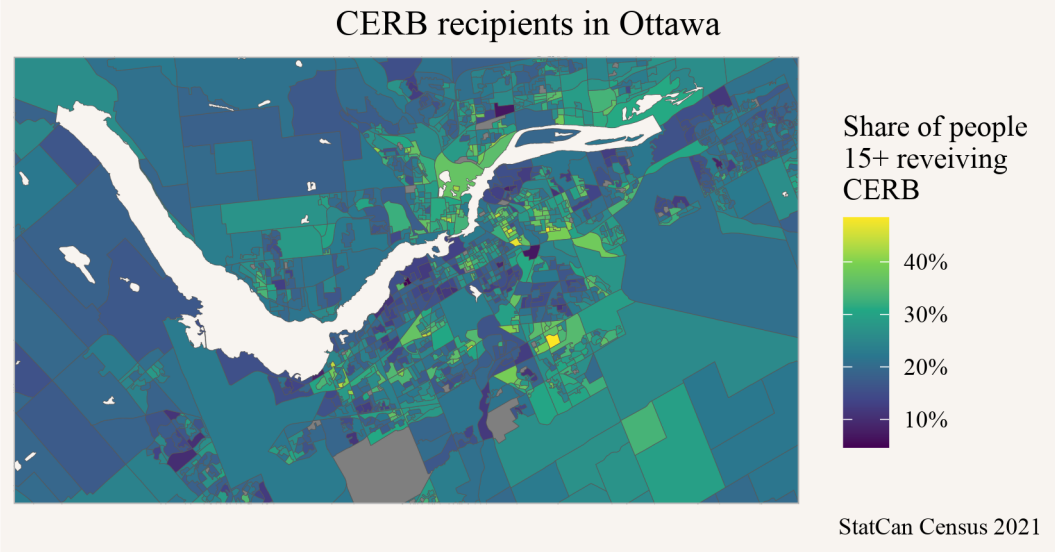
To do this we copy the code from above and paste it into a single pip, from data acquisition (using the CMA 505 for Ottawa CMA) and cut the region to the central parts by looking at the grid from the first map.

```
get_census("CA21",regions=list(CMA="505"),
                          vectors=c(cerb="v_CA21_593", base="v_CA21_554"),
                          level="CT", geo_format="sf") |>
  mutate(Share=cerb/base) |>
  ggplot() +
  geom_sf(aes(fill=Share), size=0.1) +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA, xlim=c(-76,-75.5), ylim=c(45.3,45.5)) +
  labs(title="CERB recipients in Ottawa",
       fill="Share of people\n15+ reveiving\nCERB",
       caption="StatCan Census 2021")
```



This brings out the central regions much better. We could also try this with finer geographies, setting the level to dissemination areas instead of census tracts. The same code as before works, except changing the `level="CT"` to `level="DA"`.

```
get_census("CA21",regions=list(CMA="505"),
                          vectors=c(cerb="v_CA21_593", base="v_CA21_554"),
                          level="DA", geo_format="sf") |>
  mutate(Share=cerb/base) |>
  ggplot() +
  geom_sf(aes(fill=Share), size=0.1) +
  scale_fill_viridis_c(labels=scales::percent) +
  coord_sf(datum=NA, xlim=c(-76,-75.5), ylim=c(45.3,45.5)) +
  labs(title="CERB recipients in Ottawa",
       fill="Share of people\n15+ reveiving\nCERB",
       caption="StatCan Census 2021")
```



6.5. Interpretation

We notice substantial differences in the share of people receiving CERB benefits, with rural areas generally having lower shares, and central areas being more mixed, varying between under 10% to well over 40% of people 15 years and over receiving CERB. Generally areas with lower incomes have benefited more from CERB.

7. Cars vs SUVs in Canada

Private motor vehicles in Canada seem to be getting larger and it feels like SUVs and light trucks are taking over. This subjective feeling prompts us to ask the following question to check if this is just our imagination or a real phemominon.

7.1. Question

Are SUVs and light trucks taking over in Canada?

This question is somewhat vague, it's not clear what *taking over* means. But the question is clear enough to get us started on some descriptive analysis.

7.2. Data sources

Data discovery can be challenging, but just typing "statcan motor vehicle sales" into a search engine is a good start and gets us to the StatCan table enumerating data on new motor vehicle sales. We can also use the built-in search functionality in the {cansim} package.

```
library(dplyr)
library(ggplot2)
library(cansim)

search_cansim_cubes("motor vehicle sales") |>
  select(cansim_table_number,cubeTitleEn,cubeStartDate,cubeEndDate)
```

```
# A tibble: 2 x 4
  cansim_table_number cubeTitleEn                          cubeStar~1 cubeEndD~2
  <chr>               <chr>                                <date>     <date>
1 20-10-0001          New motor vehicle sales              1946-01-01 2022-07-01
2 20-10-0002          New motor vehicle sales, by type of~ 2010-01-01 2021-01-01
# ... with abbreviated variable names 1: cubeStartDate, 2: cubeEndDate
```

There are two tables with motor vehicle sales, we can inspect them on the web or via the {cansim} package. The second table covers a much shorter time period, and is also less recent. We will check out the first table to see if it fits our needs.

To access the web we can simply type `view_cansim_webpage("20-10-0001")` into the console, which will open the StatCan webpage for Table 20-10-0001. Getting table overview data via the {cansim} package requires to load the table first, which can be slow for larger tables.

```
get_cansim_table_overview("20-10-0001")
```

This tells us that Table 20-10-0001 might have the information we need, we check the table notes to better understand what "Trucks" entails, selecting the two columns we are interested in.

```
get_cansim_table_notes("20-10-0001") %>%
  select(`Member Name`,Note) %>%
  knitr::kable()
```

Member Name	Note
NA	Prior to 1953, data for Canadian and United States manufactured vehicles and overseas manufactured vehicles are not segregated.
British Columbia and the Territories	Includes Yukon, Northwest Territories and Nunavut.
Trucks	Trucks include minivans, sport-utility vehicles, light and heavy trucks, vans and buses.
Total, overseas	Includes Japan and other countries.
NA	Seasonally adjusted data for the New Motor Vehicle Sales survey are available for the period between January 1991 to February 2012.

It looks like "Trucks" does includes SUVs, but next to light trucks it also includes heavy trucks and buses. It also includes minivans, and thinking back at our original question, we might want to refine it to include minivans.

This allows us to separate passenger cars from basically everything else. Thinking that heavy truck and bus sales probably only make up a small portion, we could use that as a stand-in for our "SUVs and light trucks" in our question. But the match is not ideal and this leaves questions open.

Maybe Table 20-10-0002 works better for our purposes, time to look at the table overview.

```
get_cansim_table_overview("20-10-0002")
```

The frequency is only annual as opposed to the monthly data from the previous table, but the breakdown of vehicle types looks much better for our purposes, it allows us to distinguish light trucks from heavy trucks and buses. Time to check the table notes for more details on the definitions.

```
get_cansim_table_notes("20-10-0002") %>%
  select(`Member Name`,Note) %>%
  knitr::kable()
```

Member Name	Note
British Columbia and the Territories	Includes Yukon, Northwest Territories and Nunavut.
Trucks	Trucks include minivans, sport-utility vehicles, light and heavy trucks, vans and buses.
Light trucks	Light trucks: include minivans, sport-utility vehicles, light trucks and vans.
Heavy trucks	Heavy trucks: include class 4, 5, 6, 7 and 8 trucks.

This looks like it fits what we need, we want to compare unit sales of Passenger cars to Light trucks.

7.3. Data acquisition

Getting the data is easy now. The {cansim} package will automatically add a native `Date` column, to convert annual data to dates it defaults to July 1st of that year. While it is a sensible default to assign a mid-year date to annual data, later on for plotting it will be more convenient for us to set the date at January 1st, so we change override the default using the optional `default_month` argument.

```
data <- get_cansim("20-10-0002", default_month = 1)

data |> select(Date,GEO,`Vehicle type`,Sales,val_norm) %>%
  head()
```

```
# A tibble: 6 x 5
  Date       GEO    `Vehicle type`            Sales      val_norm
  <date>     <chr>  <fct>                     <fct>         <dbl>
1 2010-01-01 Canada Total, new motor vehicles Units       1584499
2 2010-01-01 Canada Total, new motor vehicles Dollars 52315609000
3 2010-01-01 Canada Passenger cars            Units        710214
4 2010-01-01 Canada Passenger cars            Dollars 18982437000
5 2010-01-01 Canada Trucks                    Units        874285
6 2010-01-01 Canada Trucks                    Dollars 33333173000
```

Quick inspection of the data, using the columns we identified in the overview, helps identify the basic structure of the data.

7.4. Data preparation

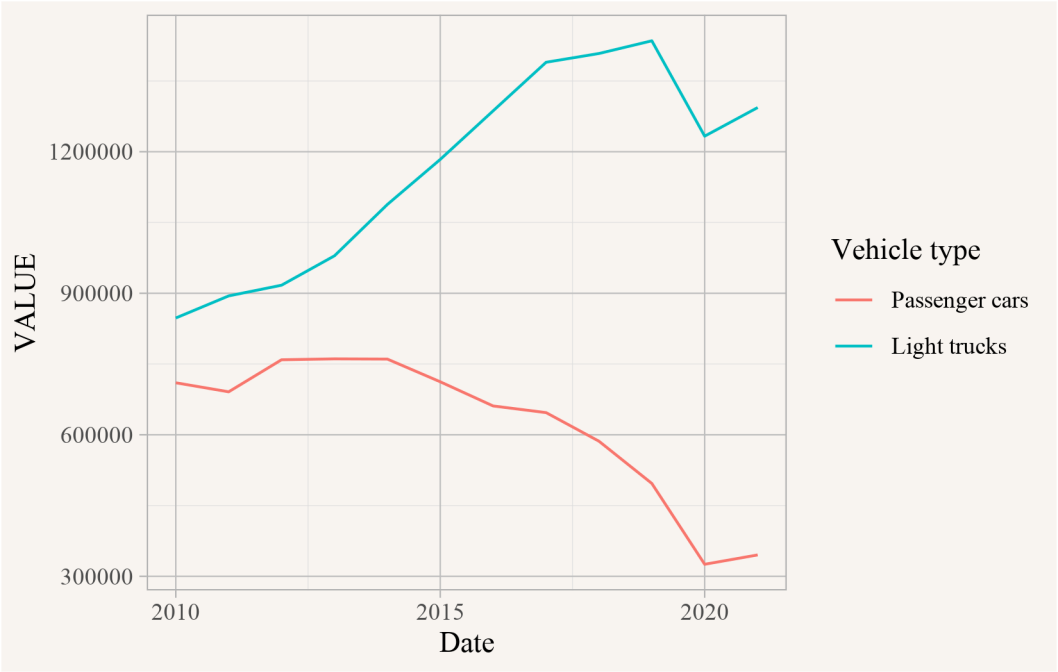
There is not much data preparation needed, we just filter down to the data we are interested in.

```
plot_data <- data |>
  filter(`Vehicle type` %in% c("Passenger cars","Light trucks"),
         Sales=="Units",
         GEO=="Canada")
```

7.5. Analysis and visualization

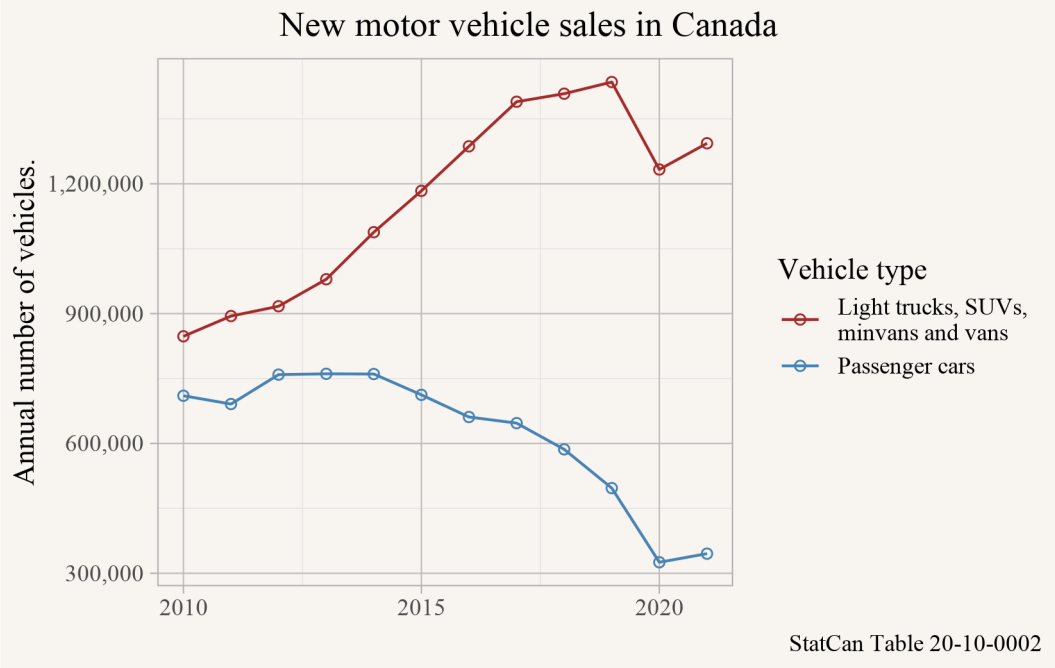
Time to lake a look what this looks like, to plot we filter for the overall Canadian data series, and tell `ggplot` to map *Date* on the x-axis, the values colum '`VALUE`` on the y-axis, and colour by vehicle type.

```
ggplot(plot_data,aes(x=Date,y=VALUE,colour=`Vehicle type`)) +
  geom_line()
```



This is looking good, time to clean up the graph a bit. We add markers for the data points, nicer axis labels, as well as a title and labels.

```
ggplot(plot_data,aes(x=Date,y=VALUE,colour=`Vehicle type`)) +
  geom_point(shape=21) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  scale_color_manual(labels=c("Light trucks"="Light trucks, SUVs,\nminvans and vans"),
                     values=c("Light trucks"="brown","Passenger cars"="steelblue")) +
  labs(title="New motor vehicle sales in Canada",
       x=NULL,y="Annual number of vehicles.",
       caption="StatCan Table 20-10-0002")
```



This answers our question in that more light trucks, SUVs, minivans and vans are sold than cars, and the gap has been growing. But the data only starts in 2010, and we suspect that things weren't always this way. At what point did SUVs and light trucks overtake new car sales?

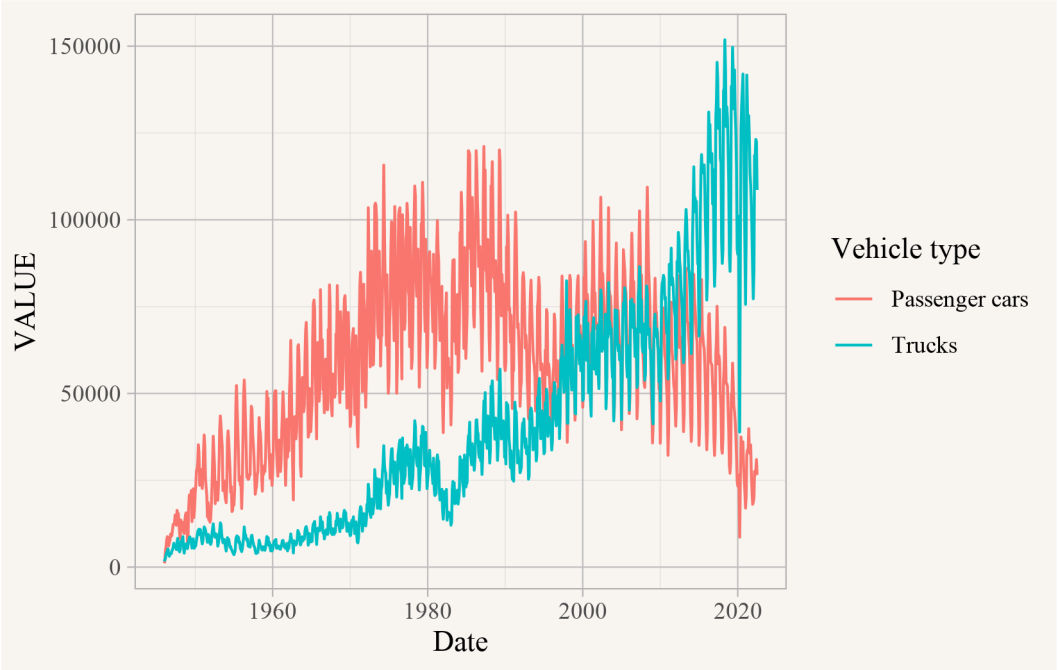
To answer than we need to jump back and load the other time series. It won't let us separate out heavy trucks and buses, but we can estimate how bad the difference is by comparing it to this data.

We load the data and filter it down to the parts that we are interested in.

```
data2 <- get_cansim("20-10-0001")

plot_data2 <- data2 |>
  filter(`Vehicle type` %in% c("Passenger cars","Trucks"),
         Sales=="Units",
         `Origin of manufacture`=="Total, country of manufacture",
         `Seasonal adjustment`=="Unadjusted",
         GEO=="Canada")
```

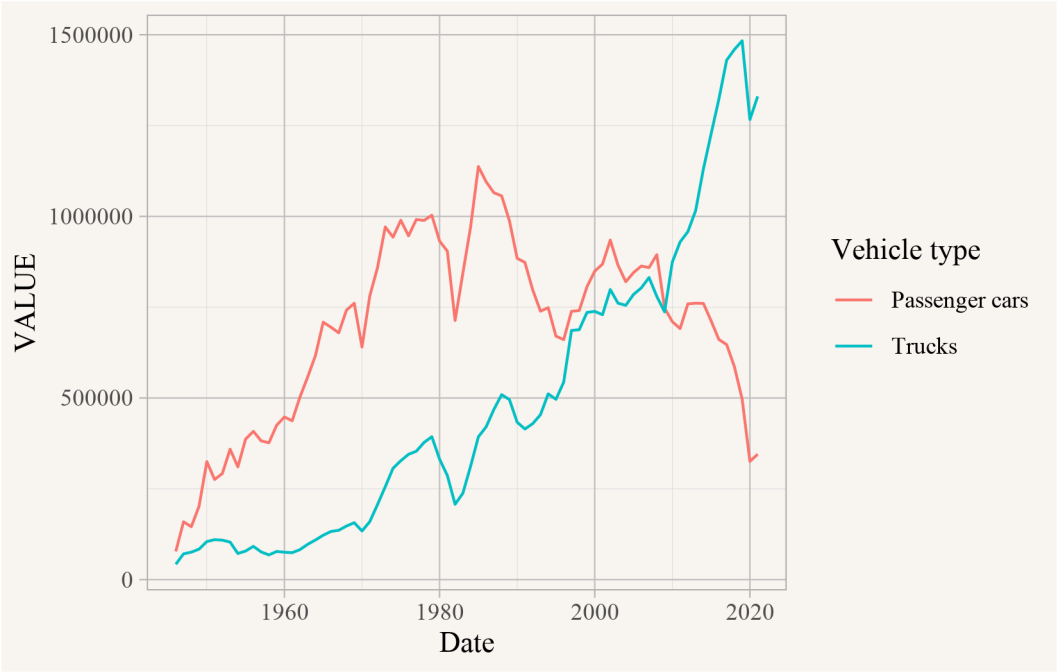
A quick plot gives us a general idea what this looks like.

```
plot_data2 %>%
  ggplot(aes(x=Date,y=VALUE,colour=`Vehicle type`)) +
  geom_line()
```



There is a strong seasonal pattern in vehicle sales, for now we will just aggregate it to annual sales so we can compare it with the previous data. For this we extract they `Year` from the `Date` column, group by `Year` and `Vehicle type` and summarize by adding up the 'VALUE' column. We added a count column to keep track how many months we added up so we can later ensure we are only showing years for which we have complete data.

```
plot_data2_annual <- plot_data2 |>
  mutate(Year=strftime(Date,"%Y")) |>
  group_by(Year,`Vehicle type`) |>
  summarise(VALUE=sum(VALUE), n=n(),.groups="drop") |>
  mutate(Date=as.Date(paste0(Year,"-01-01"))) |>
  filter(n==12)  # only use years with full 12 months of data

ggplot(plot_data2_annual,aes(x=Date,y=VALUE,colour=`Vehicle type`)) +
  geom_line()
```



Time to combine this with our previous data. This tells us that the most interesting change happened 1985 and onward, so we will discard earlier years. One quick sanity check is to see if the annual passenger car sales derived from the two series agree for the years where they are in common. Here we join the two data tables by `Date` and `Vehicle type` in order to compare the two estimate. We rename the `VALUE` column on the first one in order to avoid name conflicts.

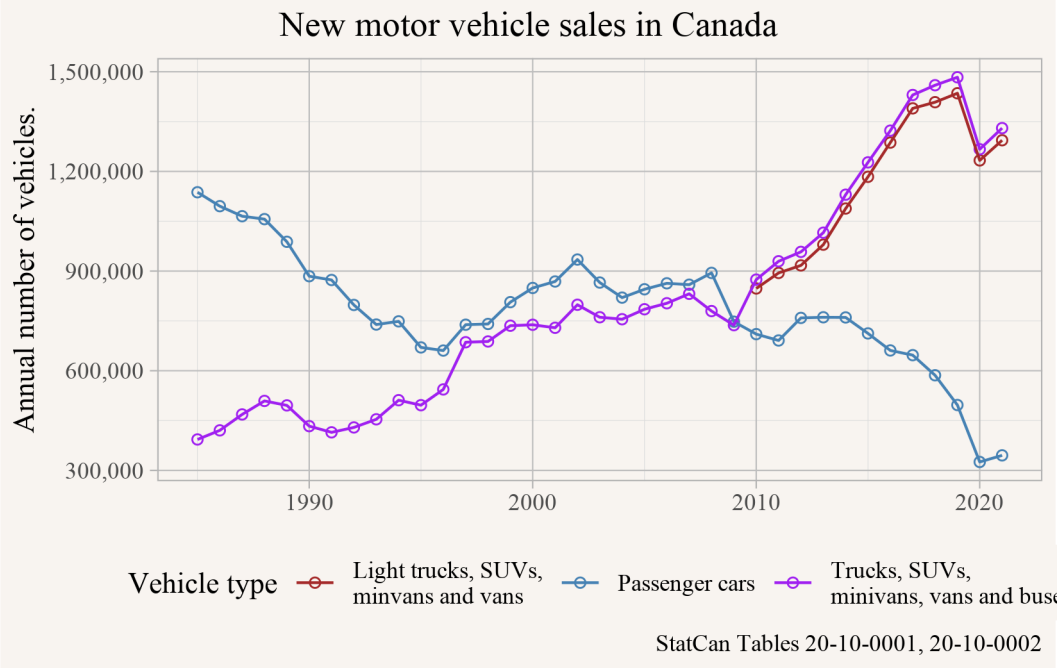
```
plot_data %>%
  filter(`Vehicle type`=="Passenger cars") %>%
  select(Date,`Vehicle type`,VALUE1=VALUE) %>%
  left_join(plot_data2_annual,by=c("Date","Vehicle type")) %>%
  mutate(diff=VALUE1-VALUE) %>%
  select(Year,VALUE1,VALUE,diff)
```

```
# A tibble: 12 x 4
   Year  VALUE1  VALUE  diff
   <chr>  <dbl>  <dbl> <dbl>
 1 2010  710214 710214     0
 2 2011  691079 691079     0
 3 2012  759024 759024     0
```

```
 4 2013  760924 760920     4
 5 2014  760449 760449     0
 6 2015  712322 712322     0
 7 2016  661088 661088     0
 8 2017  646960 646960     0
 9 2018  586357 586357     0
10 2019  496851 496851     0
11 2020  325494 325494     0
12 2021  345350 345350     0
```

The data for all years agrees, except for 2013 where one series counts 4 more passenger cars.

```
bind_rows(plot_data %>% filter(`Vehicle type`!="Passenger cars"),
          plot_data2_annual %>% filter(Date>=as.Date("1985-01-01"))) %>%
  ggplot(aes(x=Date,y=VALUE,colour=`Vehicle type`)) +
  geom_point(shape=21) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  scale_color_manual(labels=c("Light trucks"="Light trucks, SUVs,\nminvans and vans",
                              "Trucks"="Trucks, SUVs,\nminivans, vans and buses"),
                     values=c("Light trucks"="brown","Passenger cars"="steelblue",
                              "Trucks"="purple")) +
  theme(legend.position = "bottom") +
  labs(title="New motor vehicle sales in Canada",
       x=NULL,y="Annual number of vehicles.",
       caption="StatCan Tables 20-10-0001, 20-10-0002")
```



7.6. Interpretation

This confirms our initial suspicion that the "Trucks" category is dominated by light trucks, SUVs, minivans and vans, at least for the years 2010 onwards where we have data for both. Which gives us confidence to say that truck and SUV sales caught up to passenger cars sales by around 1997, and the two evolved fairly parallel until 2009, after which SUVs and light trucks increased dramatically and passenger car sales fell.

8. Under construction

Units under construction give some indication of construction activity beyond starts and completions.

How many homes are currently under construction in Toronto?

8.1. Data sources

CMHC tracks information on housing starts and completions. And the number of homes under construction, that is dwelling units that have started but aren't yet completed.

CMHC defines a housing "start" as the time when the foundation is finished, so digging a parking crater and building below ground happens before what CMHC calls a building "start". This might differ from how one might colloquially think about units under construction, as there can be significant construction activity before a "start". But this probably comes reasonably close to our question of interest.

8.2. Data acquisition

The **cmhc** package facilitates importing data from CMHC. This pries data out the Housing Market Information Portal where data is organized across a variety of tables. The easiest way to locate a table of interest is to use the `select_cmhc_table()` function from the **cmhc** package in the console to interactively step through the process. In our case, we are interested in data from the Starts and Completions Survey (**Scss**), look at the **Under Construction** series, after which we can select to have data broken down by B**edroom Type** or I**ntended Market**, where we select the former. Lastly we need to decide the breakdown type, either a level of geography or **Historical Time Periods** for a fixed geography, which is what we are interested in.

Going through this process gives us the code we need to access the data, all we need to do is fill in the geographic identifier. The **cmhc** package is designed to work in conjunction with other census data, so it uses the same geographic identifiers and translates them to CMHC's own internal geographic identifiers under the hood. For Toronto, we need to

decided if we are interested in the City of the metro area and grab the geographic identifier from the CensusMapper API tool. We will query data for the City of Toronto with standard StatCan geographic identifier "3520005" .

```
library(tidyverse)
library(cmhc)
under_construction <- get_cmhc(survey = "Scss",
                               series = "Under Construction",
                               dimension = "Dwelling Type",
                               breakdown = "Historical Time Periods",
                               geo_uid = "3520005")
```

8.3. Data preparation

There is really not much to do here, let's just inspect what the data looks like

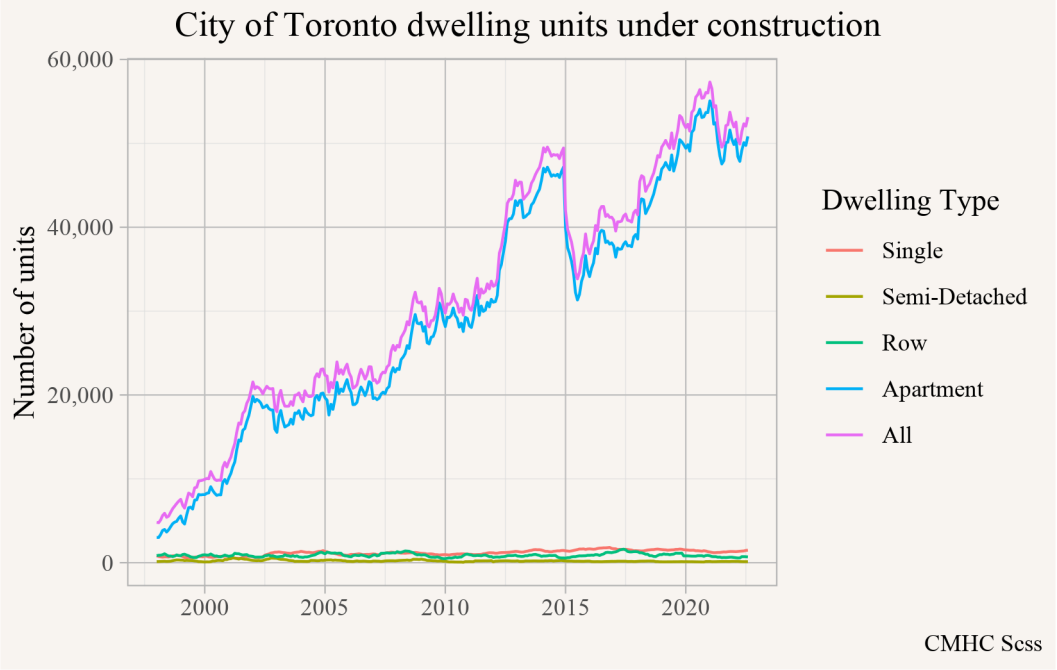
```
under_construction |> head()
```

```
# A tibble: 6 x 7
  GeoUID  Date       DateString `Dwelling Type` Value Survey Series
  <chr>   <date>     <chr>      <fct>           <dbl> <chr>  <chr>
1 3520005 1998-01-01 Jan 1998   Single            838 Scss   Under Construction
2 3520005 1998-01-01 Jan 1998   Semi-Detached     132 Scss   Under Construction
3 3520005 1998-01-01 Jan 1998   Row               838 Scss   Under Construction
4 3520005 1998-01-01 Jan 1998   Apartment        3008 Scss   Under Construction
5 3520005 1998-01-01 Jan 1998   All              4816 Scss   Under Construction
6 3520005 1998-02-01 Feb 1998   Single            738 Scss   Under Construction
```

8.4. Analysis and visualization

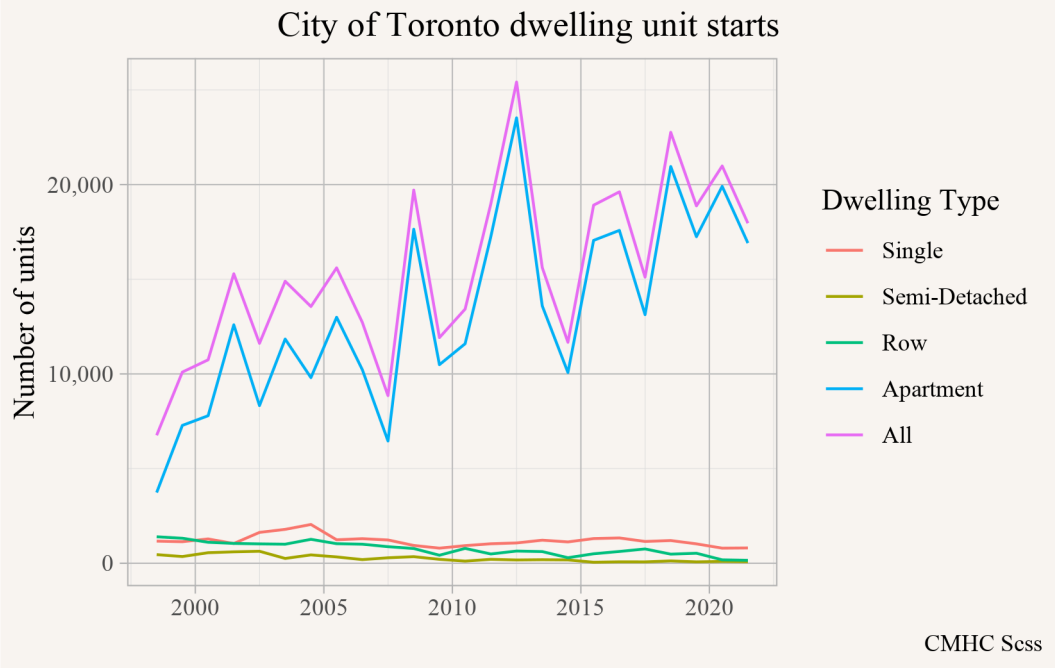
What's left is to plot the data, broken out by dwelling type.

```
ggplot(under_construction, aes(x=Date,y=Value,colour=`Dwelling Type`)) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  labs(title="City of Toronto dwelling units under construction",
       x=NULL,y="Number of units", caption="CMHC Scss")
```



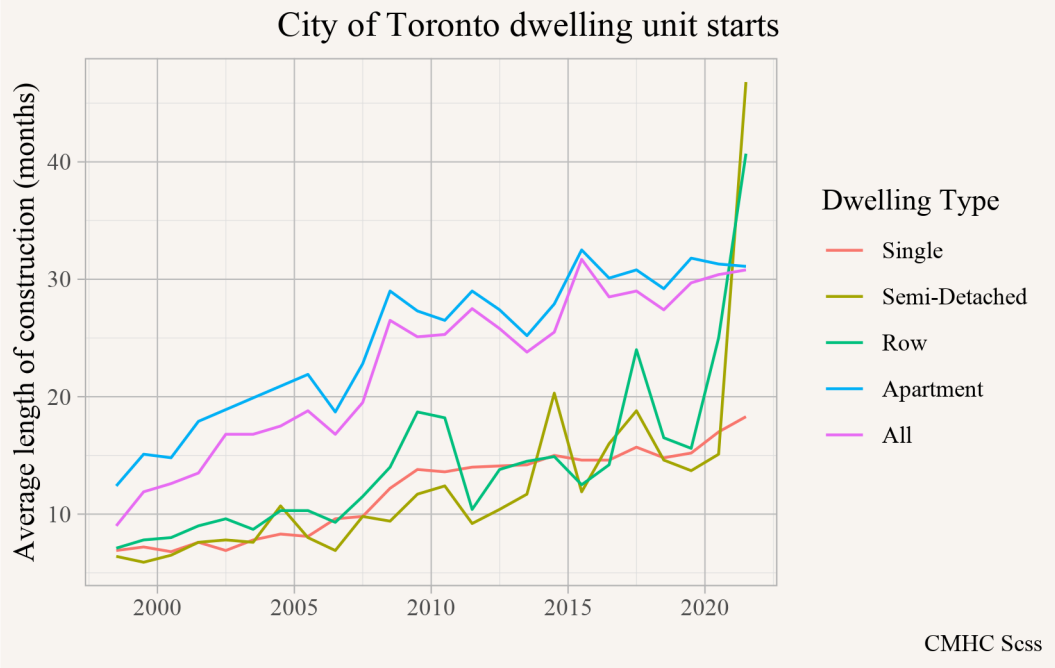
It looks like the number of units under construction, especially apartment units, has increased considerably over time. Let's cross-check that against housing starts. These tend to by quite noisy, so we go to annual frequency instead of monthly. We can adapt the code above for data acquisition and graphing into one chunk.

```
get_cmhc(survey = "Scss",
         series = "Starts",
         dimension = "Dwelling Type",
         breakdown = "Historical Time Periods",
         frequency = "Annual",
         geo_uid = "3520005") |>
  ggplot(aes(x=Date,y=Value,colour=`Dwelling Type`)) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  labs(title="City of Toronto dwelling unit starts",
       x=NULL,y="Number of units", caption="CMHC Scss")
```



Starts have increased, but not that much. Something else must be at play too, let's look at how length of construction has changed over this timeframe, again using annual data to cut down on noise.

```
get_cmhc(survey = "Scss",
         series = "Length of Construction",
         dimension = "Dwelling Type",
         breakdown = "Historical Time Periods",
         frequency = "Annual",
         geo_uid = "3520005") |>
  ggplot(aes(x=Date,y=Value,colour=`Dwelling Type`)) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  labs(title="City of Toronto dwelling unit starts",
       x=NULL,y="Average length of construction (months)",
       caption="CMHC Scss")
```



And indeed, the length of construction shot up a lot, for apartments from around 13 months in the late 90s to about 30 months around 2020. That means we now have over twice as many construction sites for the same number of units coming to market compared to the late 90s.

The sharp increase in construction time for Semi-detached and row houses might well be a data anomaly, where low and dropping number of starts of such units can be disproportionally impacted by a couple of stalled projects.

8.5. Interpretation

The units under construction has increased a lot in the City of Toronto, due to the combined effects of increasing building starts as well as a more than doubling of average time to complete these units.

9. Geography of income change

Incomes in a region change by people getting higher (or lower) incomes as well as people moving in and out of a region. We can observe the aggregate effects by looking at change in income statistics.

Where and how did incomes change in the City of Vancouver?

9.1. Data sources

The main data sources for fine-geography income data is the census, although custom tabulations of T1FF taxfiler data can offer insight of this on an annual basis at the census tract geography. For our question we are interested in broad temporal ranges, so the 5-year census data will work well.

We need to decide which income concept is best suited for our question, it is worthwhile to spend some time with the Census Income Reference Guide to understand how the data was collected and what income concept to use. Prior to 2011 the income data was part of the long form census. In 2011 the mandatory long form was replaced with the voluntary NHS, given people the option to link directly to T1FF taxfiler data or to detail the income data manually. Starting 2016 income data was linked for all people to the T1FF taxfiler data.

The question what income concept to use, e.g. individual income, household income, family income, employment income, etc, depends on the particular question we are interested in. For now we will go with family income, trying to understand how the income situation of families varies across Vancouver and across time. Family income is less affected by demographic factors like the distribution of single vs multiple person households, but is still impacted by e.g. differences in shares of seniors vs young families vs families at the peak of their earnings.

9.2. Data acquisition

We again use the CensusMapper API tool to locate the internal CensusMapper identifiers for Median Total Income of Economic Families for the years 2006 through 2021. For 2001 the standard census products reported income for census families instead of economic families, so they aren't directly comparable. As geographic breakdown we choose census tracts.

```
library(tidyverse)
regions <- list(CSD="5915022")
income_vectors <- c("2021"="v_CA21_965",
                    "2016"="v_CA16_2447",
                    "2011"="v_CA11N_2456",
                    "2006"="v_CA06_1741")
```

To facilitate the data import we write a wrapper function to acquire the census data for each of our four years. For a given census year we create the corresponding dataset identifier and select the appropriate income variable. to reduce clutter we select just the income variable and also keep the geographic identifier, and add the census year to the table.

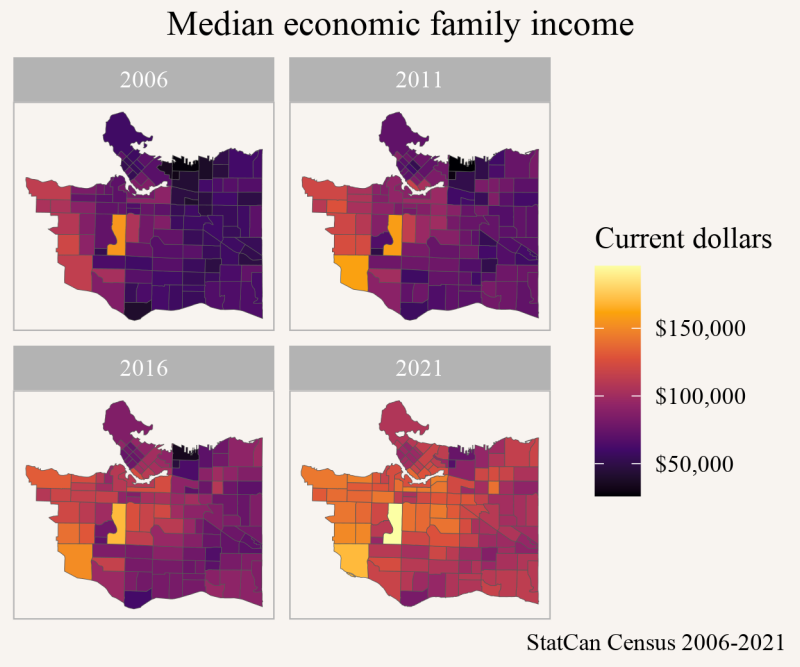
```
get_census_data <- function(year){
  year <- as.character(year)
  dataset <- paste0("CA",substr(year,3,4))
  get_census(dataset,regions=regions,
             vectors=c("ef_income"=as.character(income_vectors[year])),
             geo_format="sf",level="CT") |>
    select(GeoUID,ef_income) |>
    mutate(Year=year)
}
```

Importing the data is easy now, we just call our function for each census year and collect it into a data frame.

```
library(cancensus)
income_data <- seq(2006,2021,5) |>
  map_df(get_census_data)
```

Let's take a quick look.

```
ggplot(income_data) +
  geom_sf(aes(fill=ef_income),size=0.1) +
  scale_fill_viridis_c(option="inferno", labels=scales::dollar) +
  facet_wrap(~Year) +
  coord_sf(datum=NA) +
  labs(title="Median economic family income",
       fill="Current dollars",
       caption="StatCan Census 2006-2021")
```



9.3. Data preparation

Looking at the above graph we can see the geographic variation in each year, but it is difficult to discern geographic trends over time as incomes have gone up a lot during this timeframe. It makes sense to look at inflation-adjusted incomes instead. For this we use annual consumer price index data from StatCan Table 18-10-0005. To simplify things we locate the specific vector **v41693271** for the all-time CPI.

```
library(cansim)
inflation <- get_cansim_vector("v41693271") |>
  mutate(Year=strftime(Date,"%Y")) |>
  select(Year,CPI=val_norm) |>
  filter(Year %in% names(income_vectors))

inflation
```

```
# A tibble: 4 x 2
  Year    CPI
  <chr> <dbl>
1 2006   109.
2 2011   120.
3 2016   128.
4 2021   142.
```

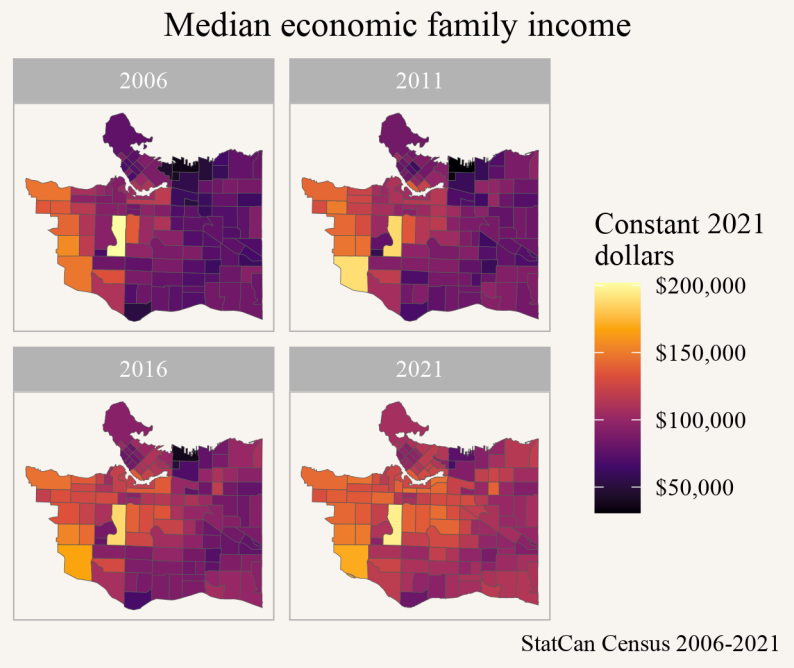
9.4. Analysis and visualization

With this, we can adjust the census data by inflation. We choose to base everything on 2021 dollars.

```
inflation <- inflation |>
  mutate(CPI=CPI/last(CPI,order_by = Year))
```

Now we just join the inflation data onto our income data by year, this adds the `CPI` column from the inflation data farme to our income with the CPI value corresponding to the value in the `Year` column in each of the two data frames. We then colour by inflation-adjusted income using the same code for graphing as above.

```
income_data |>
  left_join(inflation,by="Year") |>
  ggplot() +
  geom_sf(aes(fill=ef_income/CPI),size=0.1) +
  scale_fill_viridis_c(option="inferno", labels=scales::dollar) +
  facet_wrap(~Year) +
  coord_sf(datum=NA) +
  labs(title="Median economic family income",
       fill="Constant 2021\ndollars",
       caption="StatCan Census 2006-2021")
```



This shows more clearly how incomes have increased over time, but it would be nice to compute the change in income 2006 to 2021 for each individual census tract. But keen observers will notice that some census tracts have changed over the years, making it very difficult to compare data directly.

9.5. Data acquisition (part 2)

Fortunately the problem of making census data comparable across time has been solved with the tongfen package. This allows us to create a semi-custom tabulation on the fly on a harmonized geography based on census tracts by aggregating census data appropriately. One problem is that medians can't be aggregated, so we need to either use average income instead or be content that medians can only be approximated. By default the **tongfen** package aggregates medians as if they were averages and emits a warning. This is the route we will take for this.

To start out, we need to create metadata for the **tongfen** procedure. This is automated for Canadian census data, leveraging the metadata built into CensusMapper.

```
library(tongfen)
meta <- meta_for_ca_census_vectors(income_vectors)

meta
```

```
# A tibble: 8 x 10
  variable     label      dataset type  aggre~1 units rule  parent geo_d~2  year
  <chr>        <chr>      <chr>   <chr> <chr>   <chr> <chr> <chr>  <chr>   <int>
1 v_CA21_965   2021       CA21    Orig~ Median~ Curr~ Medi~ v_CA2~ CA21     2021
2 v_CA16_2447  2016       CA16    Orig~ Median~ Curr~ Medi~ v_CA1~ CA16     2016
3 v_CA11N_2456 2011       CA11N   Orig~ Median~ Curr~ Medi~ v_CA1~ CA11     2011
4 v_CA06_1741  2006       CA06    Orig~ Median~ Curr~ Medi~ v_CA0~ CA06     2006
5 v_CA21_964   v_CA21_964 CA21    Extra Additi~ <NA>  Addi~ <NA>   CA21     2021
6 v_CA16_2446  v_CA16_24~ CA16    Extra Additi~ <NA>  Addi~ <NA>   CA16     2016
7 v_CA11N_2455 v_CA11N_2~ CA11N   Extra Additi~ <NA>  Addi~ <NA>   CA11     2011
8 v_CA06_1729  v_CA06_17~ CA06    Extra Additi~ <NA>  Addi~ <NA>   CA06     2006
# ... with abbreviated variable names 1: aggregation, 2: geo_dataset
```

The metadata contains our original income data, as well as extra variables needed to properly aggregate the data. Getting the income data on a common geography is easy now.

```
unified_income_data <- get_tongfen_ca_census(regions,meta)
```

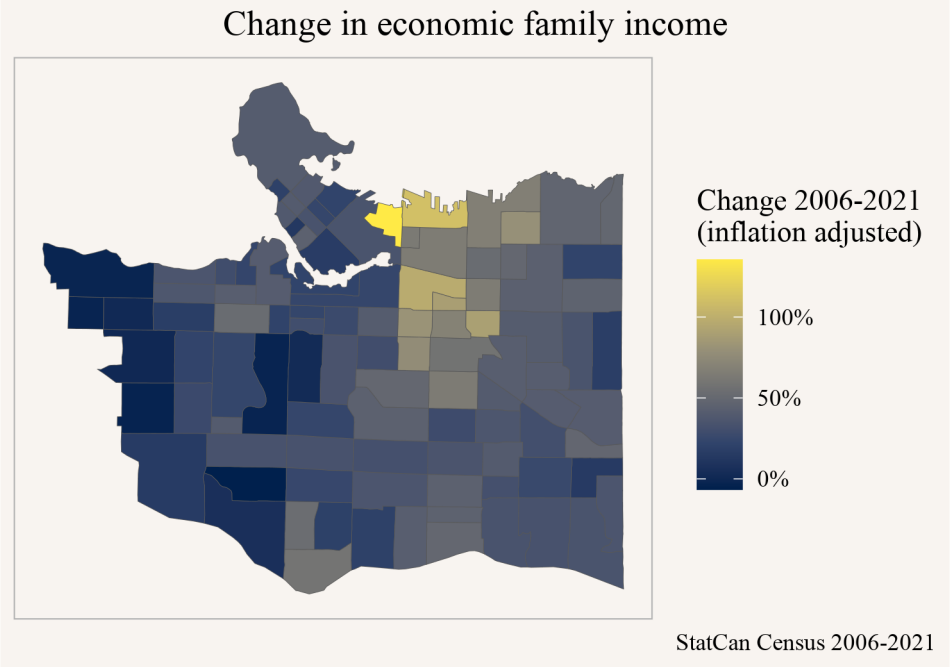
9.6. Analysis and visualization

In line with what we did before we want look at inflation-adjusted income change. To this end we extract the adjustment factor for the 2006-2021 timeframe.

```
inflation_2006_2021 <- inflation |>
  filter(Year=="2006") |>
  pull(CPI)
```

With that we can simply plot the data, mapping the inflation-adjusted percent change 2006 to 2021.

```
unified_income_data |>
  ggplot() +
  geom_sf(aes(fill=`2021`/`2006`*inflation_2006_2021-1),size=0.1) +
  scale_fill_viridis_c(option="cividis", labels=scales::percent) +
  coord_sf(datum=NA) +
  labs(title="Change in economic family income",
       fill="Change 2006-2021\n(inflation adjusted)",
       caption="StatCan Census 2006-2021")
```



9.7. Interpretation

In summary we see that income of economic families changed fasted in the Downtown Eastside, Grandview-Woodlands and Strathcona neighbourhoods, effectively doubling. Incomes increased least on the West Side, where they were already quite high to start with, and increased by about 50% throughout much of the East Side.

Part III.

Advanced descriptive analysis

Building on the section of basic descriptive analysis we will move into more advanced data processing and descriptive analysis. This will involve mixing of different datasets to tease out finer aspects. We will learn how to group and summarize data, and how to use joins.

10. BC migration

This example is motivated by a BC government press release titled "**B.C. welcomes more than 100,000 people – the most in 60 years**". This is the type of attention-grabbing headline where our gut reaction usually is to question if this is true.

Let's first try and understand what the headline really means. B.C. "welcoming" people refers to people moving to the province from elsewhere, either from other provinces or internationally. So this is referring to gross in-migration. But reading the text of the press release it immediately pivots to a different concept, saying that "B.C.'s net migration reached 100,797 people in 2021". It helpfully explains that net migration is the difference between people moving here and people moving away. Which is quite different from the number of people B.C. "welcomed" that year, or the number of people "moving to the province in 2021" as implied by the title and the first sentence of the press release.

So here comes the first difficulty, the press release is contradicting itself by mixing two concepts. That leads us to formulate a fairly broad question that should help clear this up.

10.1. Question

How many people has B.C. welcomed, net and gross, how has that changed over the last 6 decades, and how should this be interpreted?

10.2. Data sources

To start, let's figure out where that data point comes from.

The press release references StatCan as the source, let's search through the StatCan tables. Google usually works reasonably well, but we can also search programmatically. We are looking for migration estimates from the quarterly demographic estimates to get the most up-to-data population estimates from StatCan. For results we just need the first two columns, that table number and the title.

```
library(tidyverse)
library(cansim)

search_cansim_cubes("migration") |>
  filter(grepl("quarterly",cubeTitleEn)) |>
  arrange(desc(cubeEndDate)) |>
  select(1:2)
```

```
# A tibble: 2 x 2
  cansim_table_number cubeTitleEn
  <chr>               <chr>
1 17-10-0020          Estimates of the components of interprovincial migration,~
2 17-10-0040          Estimates of the components of international migration, q~
```

It looks like Table 17-10-0020 and 17-10-0040 are what we are looking for. Let's load in the data and inspect the first couple of rows for BC.

10.3. Data acquisition

```
interprovincial <- get_cansim("17-10-0020")
international <- get_cansim("17-10-0040")

interprovincial |>
  filter(GEO=="British Columbia") |>
  select(GEO,Date,`Interprovincial migration`,val_norm) |>
  tail()
```

```
# A tibble: 6 x 4
  GEO              Date       `Interprovincial migration` val_norm
  <chr>            <date>     <fct>                         <dbl>
1 British Columbia 2021-10-01 In-migrants                   12183
2 British Columbia 2021-10-01 Out-migrants                   8987
3 British Columbia 2022-01-01 In-migrants                   15970
4 British Columbia 2022-01-01 Out-migrants                  13154
5 British Columbia 2022-04-01 In-migrants                   29347
6 British Columbia 2022-04-01 Out-migrants                  25053
```

For inter-provincial migration we get in and out migration counts for every quarter. Let's also inspect the international migration data.

```
international |>
  filter(GEO=="British Columbia") |>
  select(GEO,Date,`Components of population growth`,val_norm) |>
  tail()
```

```
# A tibble: 6 x 4
  GEO              Date       `Components of population growth` val_norm
  <chr>            <date>     <fct>                                <dbl>
1 British Columbia 2022-01-01 Net non-permanent residents           3344
2 British Columbia 2022-04-01 Immigrants                           15990
3 British Columbia 2022-04-01 Emigrants                             1959
4 British Columbia 2022-04-01 Returning emigrants                   1676
5 British Columbia 2022-04-01 Net temporary emigrants               1294
6 British Columbia 2022-04-01 Net non-permanent residents          26942
```

Here we get immigrants, emigrants, returning emigrants, but for temporary emigrants and non-permanent residents we only get net change. That puts a bit of a damper on our ambition to look at gross migration, for those last two categories net is all we have.

10.4. Data preparation

Next we got to wrangle this data into a useful format. We are interested in all of these components, so we need to join these two data series together. We will retain the `GeoUID`, `GEO`, `Components of population growth`, `Date` and `val_norm` columns, which requires some renaming and then defining factor levels so that they stack nicely later in our plots. We also flip the sign on out-migrants and emigrants, as these are out-flows. To make sure those two time series start at the same time we cut it off appropriately.

The press release talked about annual change, so we do a rolling sum over 4 quarters, right-aligning the data so it's for the period of the preceding year.

```
migration_data <- bind_rows(
  interprovincial |>
    select(GeoUID,GEO,Date,
           `Components of population growth`=`Interprovincial migration`,val_norm) |>
    mutate(`Components of population growth`=
```

```
            paste0("Interprovincial ",tolower(`Components of population growth`))),
  international |>
    select(GeoUID,GEO,Date,`Components of population growth`,val_norm)
) |>
  mutate(`Components of population growth`=
           factor(`Components of population growth`,
                  levels=c("Interprovincial out-migrants",
                           "Emigrants",
                           "Interprovincial in-migrants",
                           "Immigrants",
                           "Returning emigrants",
                           "Net temporary emigrants",
                           "Net non-permanent residents"))) |>
  mutate(value=ifelse(`Components of population growth` %in%
                        c("Interprovincial out-migrants","Emigrants"),
                      -val_norm,val_norm)) |>
  filter(Date>=pmax(min(interprovincial$Date),min(international$Date))) |>
  group_by(GeoUID,`Components of population growth`) |>
  arrange(Date) |>
  mutate(annual=zoo::rollsum(value,k=4,na.pad = TRUE,align = "right")) |>
  filter(!is.na(annual)) |>
  ungroup()
```

We will also need net migration stats, so let's compute these by summing of the components,

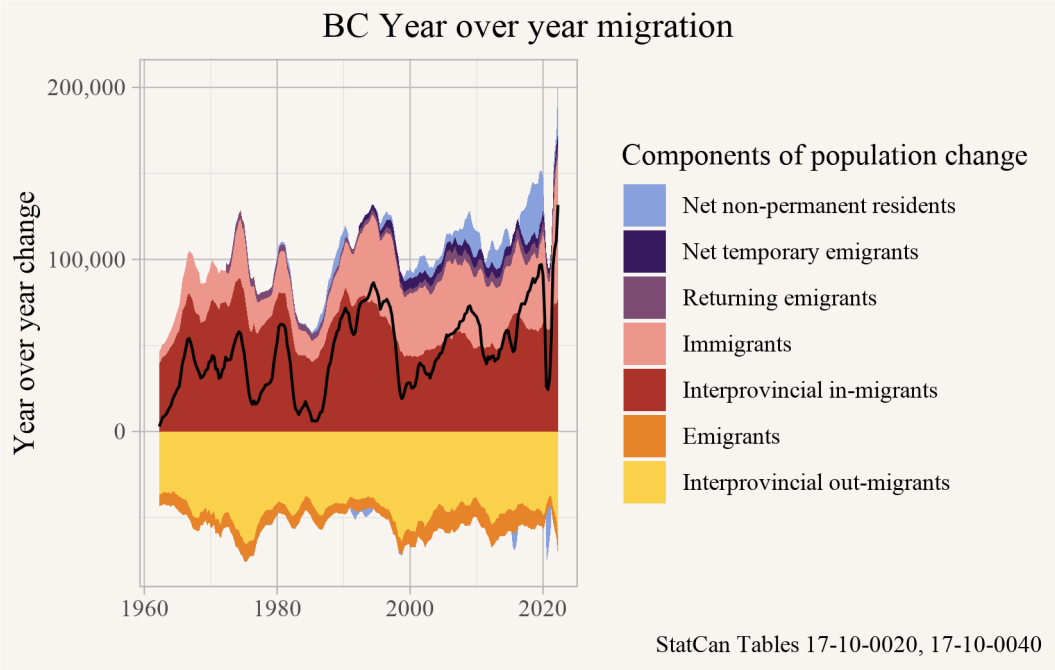
```
net_migration <- migration_data |>
  group_by(Date,GEO,GeoUID) |>
  summarize(value=sum(value),annual=sum(annual),.groups="drop") |>
  mutate(`Components of population growth`="Net migration")
```

10.5. Analysis and visualization

Time to make a graph.

```
migration_colours <- setNames(MetBrewer::met.brewer("Archambault",7),
                              migration_data$`Components of population growth` %>%
                                levels %>% rev)
```

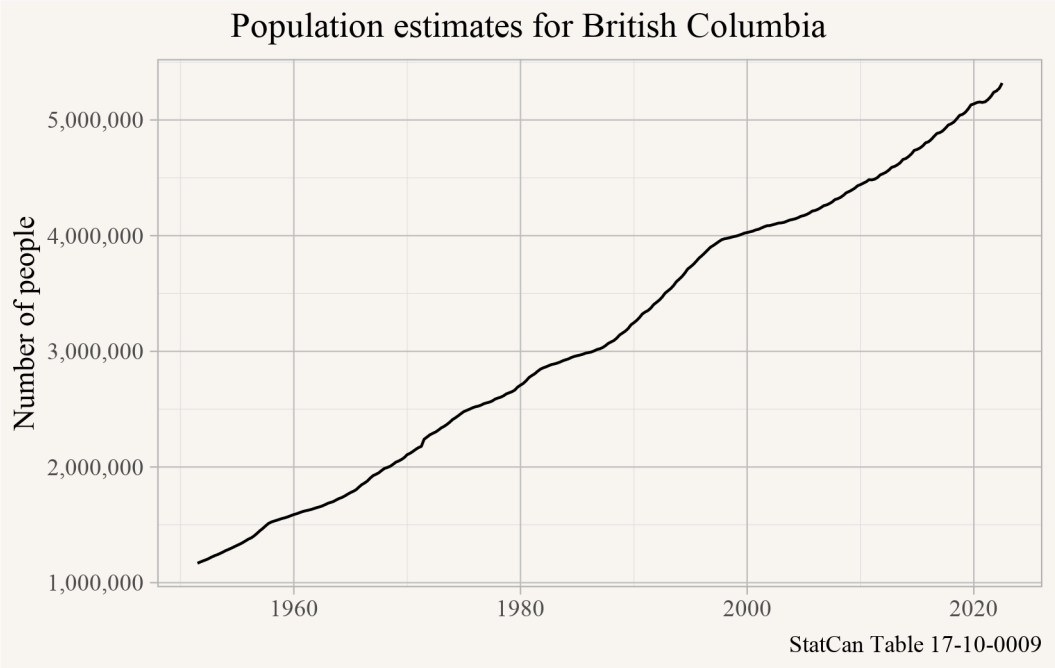
```
ggplot(migration_data |> filter(GEO=="British Columbia")) +
  geom_area(aes(x=Date,y=annual,fill=fct_rev(`Components of population growth`)),
            stat="identity") +
  scale_y_continuous(labels=scales::comma) +
  geom_line(data=net_migration |> filter(GEO=="British Columbia"),
            aes(x=Date,y=annual)) +
  scale_fill_manual(values=migration_colours) +
  labs(title="BC Year over year migration",
       y="Year over year change",x=NULL,fill="Components of population change",
       caption="StatCan Tables 17-10-0020, 17-10-0040")
```



This shows us that the press report was most likely talking did not mean to talk about number of people B.C. has "welcomed" or that "moved to the province" but instead the difference between the number of people it welcomed and the number of people it bid farewell.

And the net migration is indeed at record levels. At least in absolute terms. But B.C. now is very different from B.C. in the 60s at the start of this time series. How can we compare net migration over time in a more meaningful way? Normalizing by population is a good option here. Let's grab the data and take a look how B.C. population has changed.

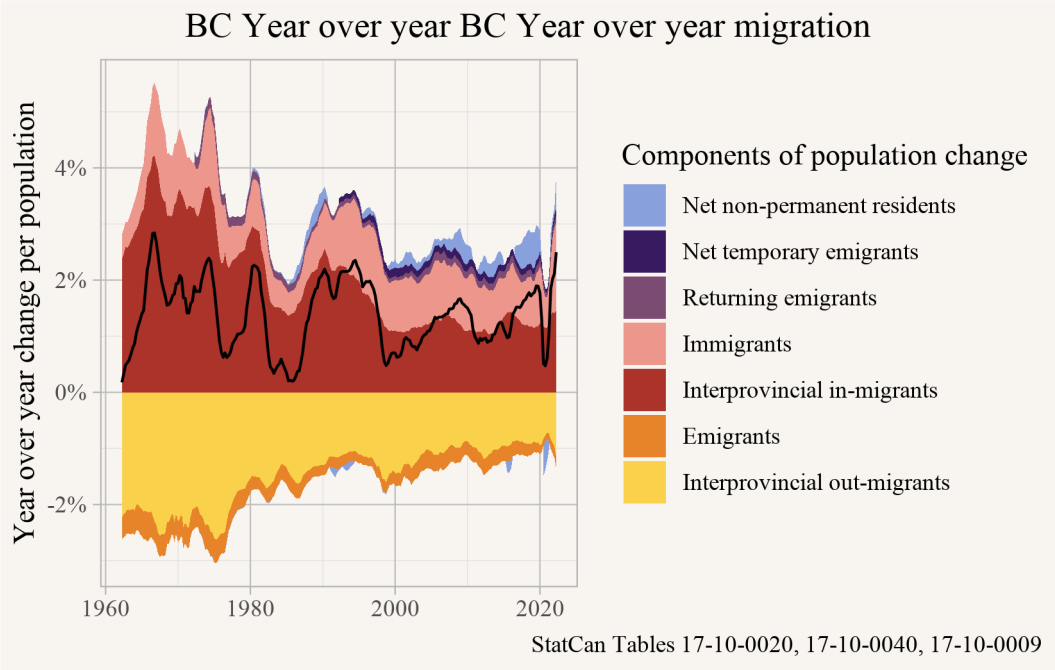
```
pop_data <- get_cansim("17-10-0009") |>
  select(GEO,Date,Population=val_norm)

pop_data |>
  filter(GEO=="British Columbia") |>
  ggplot(aes(x=Date,y=Population)) +
  geom_line() +
  scale_y_continuous(labels=scales::comma) +
  labs(title="Population estimates for British Columbia",
       y="Number of people",
       x=NULL,
       caption="StatCan Table 17-10-0009")
```



Indeed, the trend is quite strong. Let's fold that in and normalize by population.

```
migration_data |>
  left_join(pop_data, by=c("GEO","Date")) |>
  filter(GEO=="British Columbia") |>
  ggplot() +
```

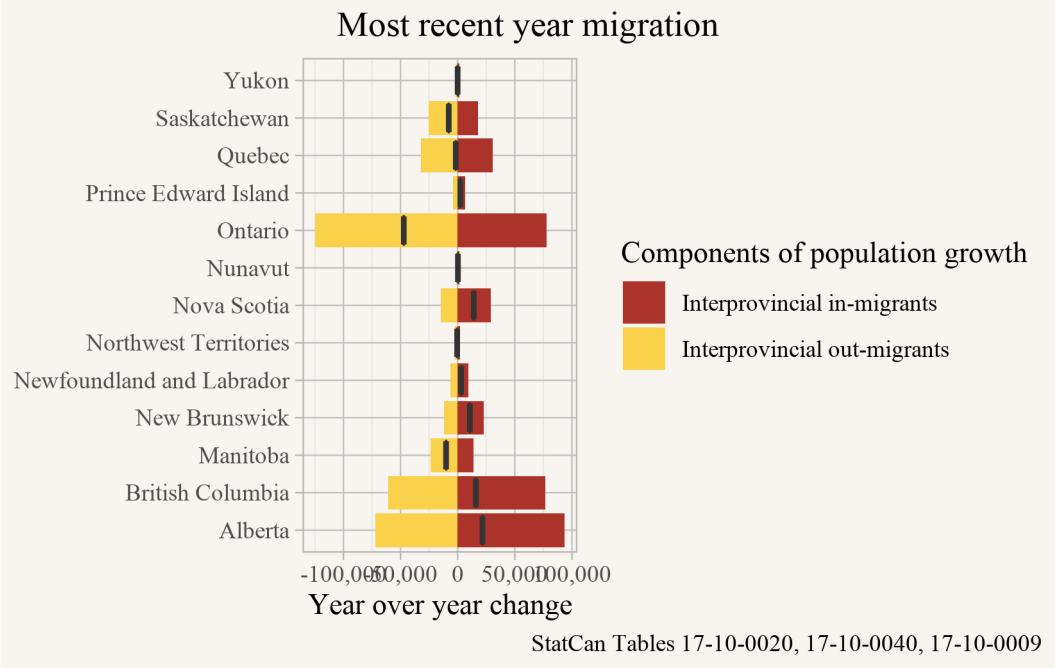
```
geom_area(aes(x=Date,y=annual/Population,fill=fct_rev(`Components of population growth`)),
          stat="identity") +
scale_y_continuous(labels=scales::percent) +
geom_line(data=net_migration |>
            left_join(pop_data, by=c("GEO","Date")) |>
            filter(GEO=="British Columbia"),
          aes(x=Date,y=annual/Population)) +
scale_fill_manual(values=migration_colours) +
labs(title="BC Year over year BC Year over year migration",
     y="Year over year change per population",x=NULL,
     fill="Components of population change",
     caption="StatCan Tables 17-10-0020, 17-10-0040, 17-10-0009")
```



Here the picture looks a little different. Net migration per capita is at its highest since the 90s, but the past 60 years there were several periods where it was larger.

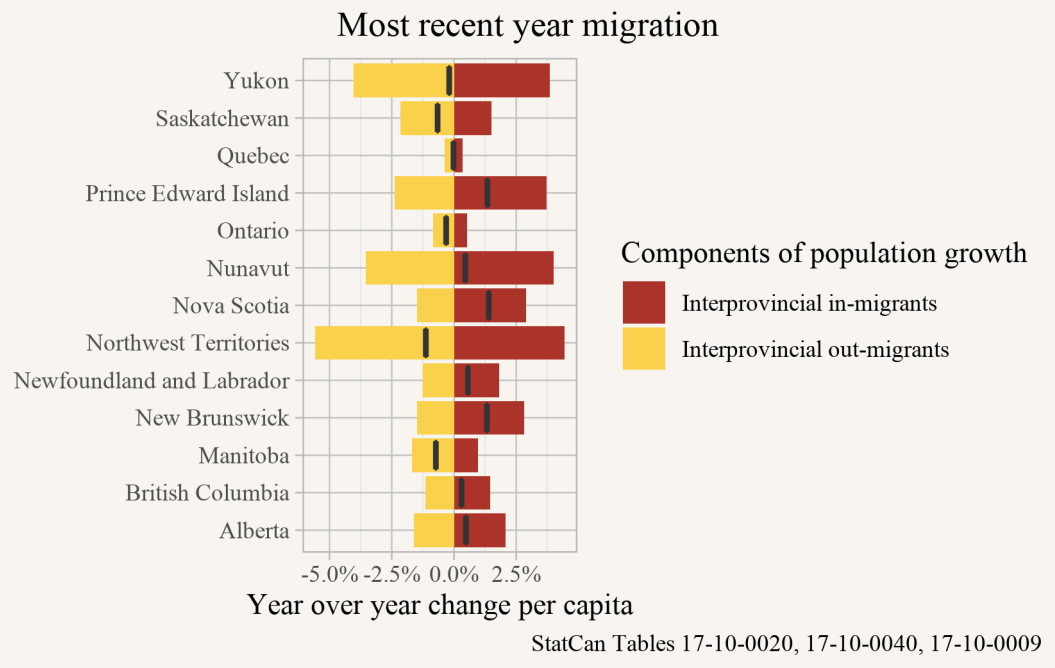
The press report also mentioned that B.C.'s interprovincial migration numbers are higher than any other province. This is easy to check now.

```
migration_data_interprovinicial <- migration_data |>
  left_join(pop_data, by=c("GEO","Date")) |>
  filter(grepl("Interprovincial",`Components of population growth`))

net_interprovincial <- migration_data_interprovinicial |>
  group_by(GEO,Date) |>
  summarize(value=sum(value),
            annual=sum(annual),
            Population=first(Population),
            .groups="drop")

migration_data_interprovinicial |>
  filter(GEO!="Canada") |>
  filter(Date==max(Date)) |>
  ggplot(aes(y=GEO,x=annual)) +
  geom_bar(stat="identity",
           aes(fill=fct_rev(`Components of population growth`))) +
  geom_boxplot(data=net_interprovincial |>
                 filter(GEO!="Canada") |>
                 filter(Date==max(Date))) +
  scale_fill_manual(values=migration_colours[grepl("Interprov",names(migration_colours))])
  scale_x_continuous(labels=scales::comma) +
  labs(title="Most recent year migration",
       fill="Components of population growth",
       y=NULL,x="Year over year change",
       caption="StatCan Tables 17-10-0020, 17-10-0040, 17-10-0009")
```



In absolute number B.C. indeed has both the highest interprovincial in-migration and interprovincial net-migration among all provinces. But the provinces have vastly different sizes, so that's not really a fair comparison. Again, we can normalize by population.

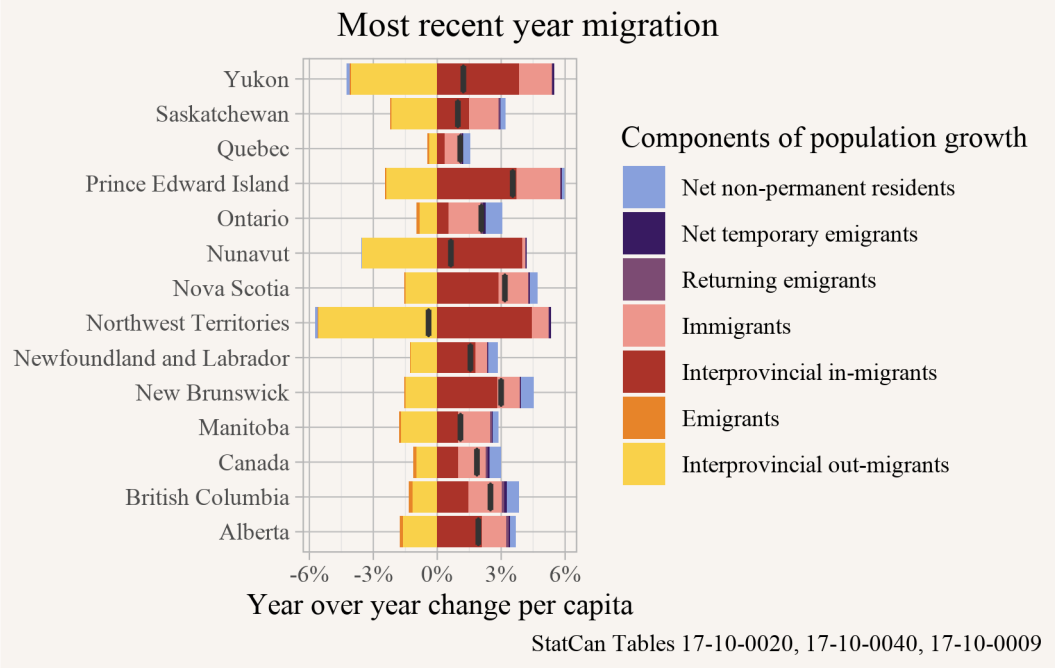
```
migration_data_interprovinicial |>
  filter(GEO!="Canada") |>
  filter(Date==max(Date)) |>
  ggplot(aes(y=GEO,x=annual/Population)) +
  geom_bar(stat="identity",
           aes(fill=fct_rev(`Components of population growth`))) +
  geom_boxplot(data=net_interprovincial |>
                 filter(GEO!="Canada") |>
                 filter(Date==max(Date))) +
  scale_fill_manual(values=migration_colours[grepl("Interprov",names(migration_colours))])
  scale_x_continuous(labels=scales::percent) +
  labs(title="Most recent year migration",
       fill="Components of population growth",
       y=NULL,x="Year over year change per capita",
       caption="StatCan Tables 17-10-0020, 17-10-0040, 17-10-0009")
```



Viewed this way B.C.'s interprovincial in-migration and net migration still looks good, but many of the other provinces beat out that growth rate.

For completeness we can also just show the full graph that includes the international migration components.

```
migration_data |>
  left_join(pop_data, by=c("GEO","Date")) |>
  filter(Date==max(Date)) |>
  ggplot(aes(y=GEO,x=annual/Population)) +
  geom_bar(stat="identity",aes(fill=fct_rev(`Components of population growth`))) +
  geom_boxplot(data=net_migration |>
                 left_join(pop_data, by=c("GEO","Date")) |>
                 filter(Date==max(Date))) +
  scale_fill_manual(values=migration_colours) +
  scale_x_continuous(labels=scales::percent) +
  labs(title="Most recent year migration",
       fill="Components of population growth",
       y=NULL,x="Year over year change per capita",
       caption="StatCan Tables 17-10-0020, 17-10-0040, 17-10-0009")
```



10.6. Interpretation

This answers our question, the latest annual net migration edges over the 100,000 people mark, and in absolute terms this is the highest it's been over at least 60 years. And B.C.'s interprovincial (net) in-migration was the highest in Canada in absolute terms. But what can we learn from that?

B.C. 60 years ago is very different from B.C. today. To account for that we can normalize by population, and the relative net migration has been higher at several times during the past 60 years, most recently in the 90s.

We also note that big dip in net-migration during COVID-19. It is not clear if the current heights are a bounce-back to make up for the comparatively low net in-migration during the pandemic, or if it is simply reverting back to the increasing trend we have seen over the past 10 years.

References