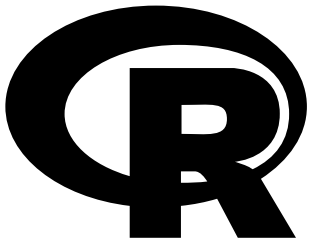
CuRso de R con MaRtín: 7a edición.



Dr. Martín Lozano.

<martin.lozano@udem.edu>

https://sites.google.com/site/mlozanoqf/

https://github.com/mlozanoqf/

agosto 21, 2021. 03:54:54

Abstract

This is an introductory applied tutorial suitable for undergraduate students interested in the area of quantitative finance modeling, data science, and its application into business. We extend examples and codes freely available in Internet, and some of my own. As in the philosophy of Donald Knuth [Knuth, 1984], the objective of this document is to explain to human beings what we want a computer to do as literate programming. This is a work in progress and it is under revision.

Index.

1 Introduction.

1.1 YouTube installation guides.

As you may understand, R and RStudio versions are frequently updated and people regularly upload installation guides on YouTube. If you find a newer video for installing a newer version please share. In any case, these videos can definitely help you as a guide to install the newer version available.

- Download & Install R 3.6.3
- Download & Install RStudio Desktop 1.3.959
- Installing R and Rstudio on MacOS.
- Orientation and Setting up R (Setting up R)
- Orientation and Setting up R (Setting up RStudio)
- RStudio: A Guided Tour (by Jamison Crawford).
- How to Install RStudio (and Knit to PDF).
- RStudio: Explaining the Interface & R Markdown.
- How to Connect RStudio with Git (Github) for Cloning and Pushing Repo.

1.2 RStudio and GitHub integration plus pull request (PR).

Here, we assume you have installed R, RStudio, Git, and you have a GitHub account. This series of videos are good to understand the basics of how professionals develop code in a collaborative way.

Video 1 of 4. I recommend you to start at minute 10. Before minute 10 you can see an installation process, but there are better videos in the previous section (YouTube installation guides). Starting from minute 10 you can learn how to link RStudio with your GitHub account using the SSH RSA key which I highly recommend. You may find Samsun's voice speed a bit slow, just remember you can change the playback speed in YouTube if necessary. Installing R, R Studio, and link R Studio with Github account - Part 4 - by Samsun.

Video 2 of 4. The relevant part starts at minute 5:50. In the first part Lisa explains how to use the `{usethis}` library to link RStudio with your GitHub account, but I recommend you to follow the video 1 for that purpose. So, please watch this from minute 5:50 to the end. There is one more thing, in 8:39 she clones with HTTPS, but I recommend you cloning with SSH, the option to select SSH is just right there in the same window. In my experience, cloning with SSH is better because of security issues and because you may not be asked to log in repeatedly. Using git and GitHub in R Studio.

Video 3 of 4. Video 2 explained RStudio and GitHub integration to work alone. This video 3 explains how to work with others as collaborators (Lisa and Heather). In the context of this class, your classmate colleagues are collaborators, I am not included as a collaborator. Note that Lisa and Heather cloned with HTTPS but I recommend cloning with SSH. They show how to work with branches and pull request between collaborators. Starting from 30:40 the tutorial is somewhat less interesting, you are free to look at it, but it is not vital for our purposes. Using Git and GitHub in R (second part).

Video 4 of 4. Layla explains very well how to interact between a team of students and your professor or instructor. Note that her RStudio panes are in a very weird location but I guess you still can follow. Layla also recommends working with branches, this is a good practice. She also knits in html but I will ask you to knit in pdf most of the time. Note that she actually takes the side of the students who are interacting with an instructor which is called reviewer in GitHub. Branching and Committing from RStudio and creating a PR on GitHub.

Complementary video. This is in Spanish, it is a good video, but the previous 4 are somewhat better. She links RStudio and GitHub using the terminal, I normally follow the procedure in the first video (Samsun's video). She also clones with HTTPS, but I recommend you cloning with SSH as I explained before. RStudio y GitHub: control de versiones.

1.3 Where to start?

Consider the following specific online and free resources to start or continue learning R:

- McNulty [2021]: Chapter 2. The Basics of the R Programming Language.
- Hanck et al. [2020]: Chapter 1.2 A Very Short Introduction to R and RStudio.
- Oswald et al. [2020]: Chapter 1. Introduction to R.
- Heiss [2020]: Chapter 1. Introduction.
- De Vries and Meys [2015]: The book is available online.
- Peng [2016]: The book is available online.
- Wickham [2016]: The book is available online, this is for learning `ggplot`.
- Xie [2021], Xie et al. [2020] and Xie et al. [2021]: The books are available online, these are for learning `rmarkdown`.
- Grolemund and Wickham [2018]: The book is available online, this is for learning about the tidyverse in R.
- To understand how R Markdown works: R Markdown guidelines.
- Thiyanga Talagala: A detailed R Markdown guide.

- Bryan [2018]: The book is available online, this is for learning Git and GitHub.
- Probability, Statistics, and Data: A fresh approach using R. Foundations of Statistics with R.
- The references in the course calendar.
- As my student you have a free DataCamp account. You can find at least 151 courses and 45 projects about R.
- Swirl teaches you R programming and data science interactively, at your own pace, and right in the R console. Swirl.
- The `learnr` package makes it easy to turn any R Markdown document into an interactive tutorial. Interactive Tutorials for R.

1.4 RStudio and GitHub.

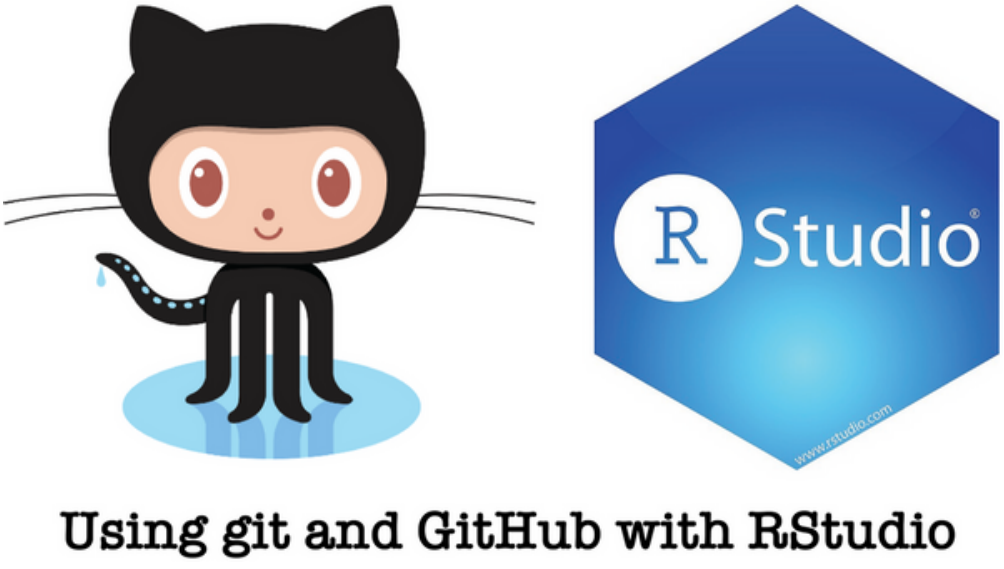


Illustration by Allison Horst.

1.5 About me.

I am currently professor of finance and economics at UDEM. I regularly collaborate with The University of Manchester (Alliance Manchester Business School); the Centre for Financial and Management Studies (CeFiMS) at SOAS University of London, among others universities.

Let me introduce myself by listing my postgraduate studies and describing my professional background.

My postgraduate studies: I have a Post Doc in Finance from The University of Manchester; a PhD in Quantitative Finance from the University of the Basque Country; a Doctor *Europaeus* mention from several European universities: Manchester Business School; University of Edinburgh; Humboldt University; Aarhus University; University of Vienna; Dublin City University; Queen Mary, University of London; and University of St. Gallen. I was a Pre-doctoral Marie Curie research fellow at The Manchester Business School. I have seven postgraduate studies in Statistical Learning, Data Mining, Scientific Analysis of Data, Statistical Methods, Applied Statistics, Finance, and Quantitative Finance. I have a BS in Economics, and other professional certificates mostly in the area of data science from Strathclyde Business School, The Alan Turing Institute, among other European institutions.

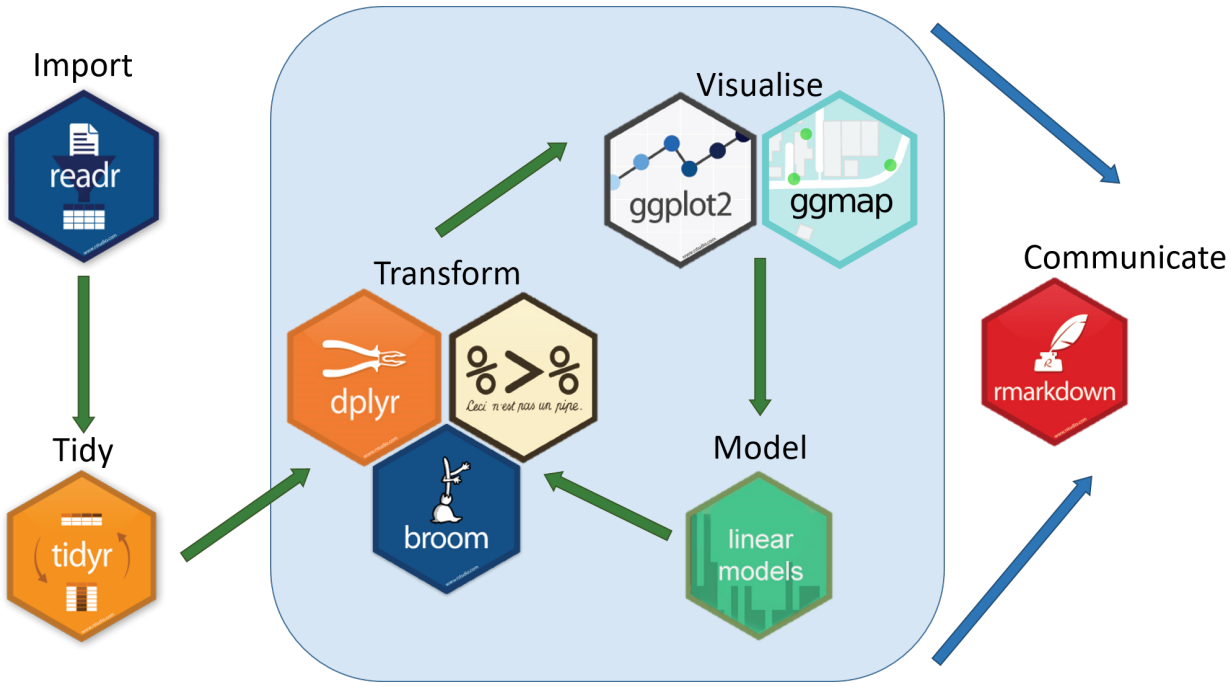
My professional background: I have been a researcher in the area of quantitative finance for the last 10 years. My research has been published in 3 stars journals (high-ranked) according to The Chartered Association of Business Schools[1], presented in numerous research seminars, and in prestigious international conferences. I have been a lecturer in economics, finance and data science for under and postgraduate levels at different universities in Mexico and the UK for the last 20 years. Also, I have supervised more than 70 dissertations at under and postgraduate academic programs of schools including the London School of Business & Finance; University of London, School of Oriental and African Studies (SOAS); The University of Manchester; Universidad Complutense de Madrid, UDEM, among others. I collaborate as reviewer and editor for several academic journals in the areas of finance and management. Also, I have experience in continuous education, consulting, and executive training in the area of finance.

My academic profile, including my full and updated curriculum vitae, can be found here:

[1]The Chartered Association of Business Schools is considered the voice of the UK's business and management education sector. According to ABS, all 3 stars rated journals publish original and well executed research papers and are highly regarded. These journals typically have good submission rates and are very selective in what they publish. Papers are heavily refereed in 3 stars rated journals.

I love art, mainly painting and music, as the expression of human creativity and imagination. I spend some free time playing my Yamaha digital piano. I also used to be an active keyboardist, piano player, and orchestra director. Most of my experience in this field has been as a musician at live concerts in Mexico and Europe, piano solo concerts, and musicals organized by the Tecnológico de Monterrey (Campus Monterrey) for about a decade.

2 Packages.



Load the relevant packages.

```
library(mxmaps)
library(viridis)
library(scales)
library(ggplot2)
library(dplyr)
library(ggrepel)
library(readr)
library(tidyquant)
library(tidyr)
library(tidyverse)
library(imager)
library(purrr)
library(meme)
library(rayshader)
```

3 Tidyverse verbs.

{dplyr} is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges. The following section is taken from Ben Stenhaug. See the following article:

https://teachingr.com/content/the-5-verbs-of-dplyr/the-5-verbs-of-dplyr-article.html

Five verbs of {dplyr}: `select()`, `filter()`, `arrange()`, `mutate()`, and `summarize()`.

Create a simple database to illustrate the value of the above verbs.

```
hamsters <- data_frame(
  name = c("Aleydis", "Carolina", "América", "Plascencia", "Martín"),
  gender = c("female", "female", "female", "male", "male"),
  hamsters = c(5, 7, 6, 2, 1),
  hamster_cages = c(2, 1, 3, 3, 4))
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 Carolina   female        7             1
## 3 América    female        6             3
## 4 Plascencia male          2             3
## 5 Martín     male          1             4
```

3.1 Arrange.

`arrange()` function keeps all of the information in the data frame, but just changes the order of the rows. This is the same thing that the *sort* feature in Excel does.

By default, arranging happens in ascending order or from low to high:

```
hamsters |>
  arrange(hamsters)
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Martín     male          1             4
## 2 Plascencia male          2             3
## 3 Aleydis    female        5             2
## 4 América    female        6             3
## 5 Carolina   female        7             1
```

We can instead arrange in descending order with the `desc()` function:

```
hamsters |>
  arrange(desc(hamster_cages))
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Martín     male          1             4
## 2 América    female        6             3
## 3 Plascencia male          2             3
## 4 Aleydis    female        5             2
## 5 Carolina   female        7             1
```

Character columns get arranged in alphabetical order:

```
hamsters |>
  arrange(name)
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 América    female        6             3
## 3 Carolina   female        7             1
## 4 Martín     male          1             4
## 5 Plascencia male          2             3
```

If we input multiple column names, `arrange()` uses the additional columns to break ties.

```
hamsters |>
  arrange(hamster_cages)
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Carolina   female        7             1
## 2 Aleydis    female        5             2
## 3 América    female        6             3
## 4 Plascencia male          2             3
## 5 Martín     male          1             4
```

```
hamsters |>
  arrange(hamster_cages, hamsters)
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Carolina   female        7             1
## 2 Aleydis    female        5             2
## 3 Plascencia male          2             3
## 4 América    female        6             3
## 5 Martín     male          1             4
```

We could be interested to use the `spread()` function to spread the gender column into two new columns (female and male).

```
spread(data = hamsters, key = gender, value = hamsters)
```

```
## # A tibble: 5 x 4
##   name       hamster_cages female  male
##   <chr>              <dbl>  <dbl> <dbl>
## 1 Aleydis                2      5    NA
## 2 América                3      6    NA
## 3 Carolina               1      7    NA
## 4 Martín                 4     NA     1
## 5 Plascencia             3     NA     2
```

3.2 Select.

`select()` function is used to choose which columns to work with. For example, maybe we want just the name and hamsters columns:

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 Carolina   female        7             1
## 3 América    female        6             3
## 4 Plascencia male          2             3
## 5 Martín     male          1             4
```

```
hamsters |>
  select(name, hamsters)
```

```
## # A tibble: 5 x 2
##   name       hamsters
##   <chr>         <dbl>
## 1 Aleydis           5
## 2 Carolina          7
## 3 América           6
## 4 Plascencia        2
## 5 Martín            1
```

We can use a `-` to get rid of a column and leave the rest of the columns:

```
hamsters |>
  select(-name)
```

```
## # A tibble: 5 x 3
##   gender hamsters hamster_cages
##   <chr>     <dbl>         <dbl>
## 1 female        5             2
## 2 female        7             1
## 3 female        6             3
## 4 male          2             3
```

```
## 5 male          1             4
```

We also could have gotten just the name and hamsters columns by removing the `gender` and `hamster_cages` columns:

```
hamsters |>
  select(-gender, -hamster_cages)
```

```
## # A tibble: 5 x 2
##   name       hamsters
##   <chr>         <dbl>
## 1 Aleydis           5
## 2 Carolina          7
## 3 América           6
## 4 Plascencia        2
## 5 Martín            1
```

`select()` can also be used to rearrange the order of columns:

```
hamsters |>
  select(hamsters, hamster_cages, gender, name)
```

```
## # A tibble: 5 x 4
##   hamsters hamster_cages gender name
##      <dbl>         <dbl> <chr>  <chr>
## 1        5             2 female Aleydis
## 2        7             1 female Carolina
## 3        6             3 female América
## 4        2             3 male   Plascencia
## 5        1             4 male   Martín
```

`everything()` is a convenient shortcut that adds all the columns that haven't been used yet. It is very useful if you want to move a column to the front of a data frame:

```
hamsters |>
  select(hamster_cages, everything())
```

```
## # A tibble: 5 x 4
##   hamster_cages name       gender hamsters
##           <dbl> <chr>      <chr>     <dbl>
## 1             2 Aleydis    female        5
```

```
## 2             1 Carolina   female        7
## 3             3 América    female        6
## 4             3 Plascencia male          2
## 5             4 Martín     male          1
```

3.3 Filter.

`filter()` function is used to select which rows you want. For example, maybe we only want students with more than 3 hamsters:

```
hamsters |>
  filter(hamsters > 3)
```

```
## # A tibble: 3 x 4
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Aleydis  female        5             2
## 2 Carolina female        7             1
## 3 América  female        6             3
```

Notice that there is a variable named the same thing as the data frame. The first **`hamsters`** in the following code refers to the data frame, while the second **`hamsters`** refers to the `hamsters` column.

Or maybe we only want female students:

```
hamsters |>
  filter(gender == "female")
```

```
## # A tibble: 3 x 4
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Aleydis  female        5             2
## 2 Carolina female        7             1
## 3 América  female        6             3
```

Notice that we had to use `==` instead of `=`. This is because `=` is for assignment (making something equal something else), whereas `==` is for comparison (seeing if two things are equal or not).

If we want to use an *and* (require that multiple conditions hold) we can either use the `&`

sign or separate the conditions with a comma. For example, the following two filters are equivalent:

```
hamsters |>
  filter(gender == "female" & hamsters >= 6)
```

```
## # A tibble: 2 x 4
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Carolina female        7             1
## 2 América  female        6             3
```

```
hamsters |>
  filter(gender == "female", hamsters >= 6)
```

```
## # A tibble: 2 x 4
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Carolina female        7             1
## 2 América  female        6             3
```

If we want to use an *or* (require that just 1 of multiple conditions holds) we have to use the | sign. For example:

```
hamsters |>
  filter(gender == "male" | hamsters >= 7)
```

```
## # A tibble: 3 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Carolina   female        7             1
## 2 Plascencia male          2             3
## 3 Martín     male          1             4
```

A case that commonly comes up is requiring that a variable has one of a set of specific values. For example, maybe we only want students with 2, 4, 6, or 8 hamsters.

The most intuitive way to do this is with a series of *or* statements:

```
hamsters |>
  filter(hamsters == 2 | hamsters == 4 | hamsters == 6 | hamsters == 8)
```

```
## # A tibble: 2 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 América    female        6             3
## 2 Plascencia male          2             3
```

It gets tedious having to type and retype the word *hamsters* over and over again, though.

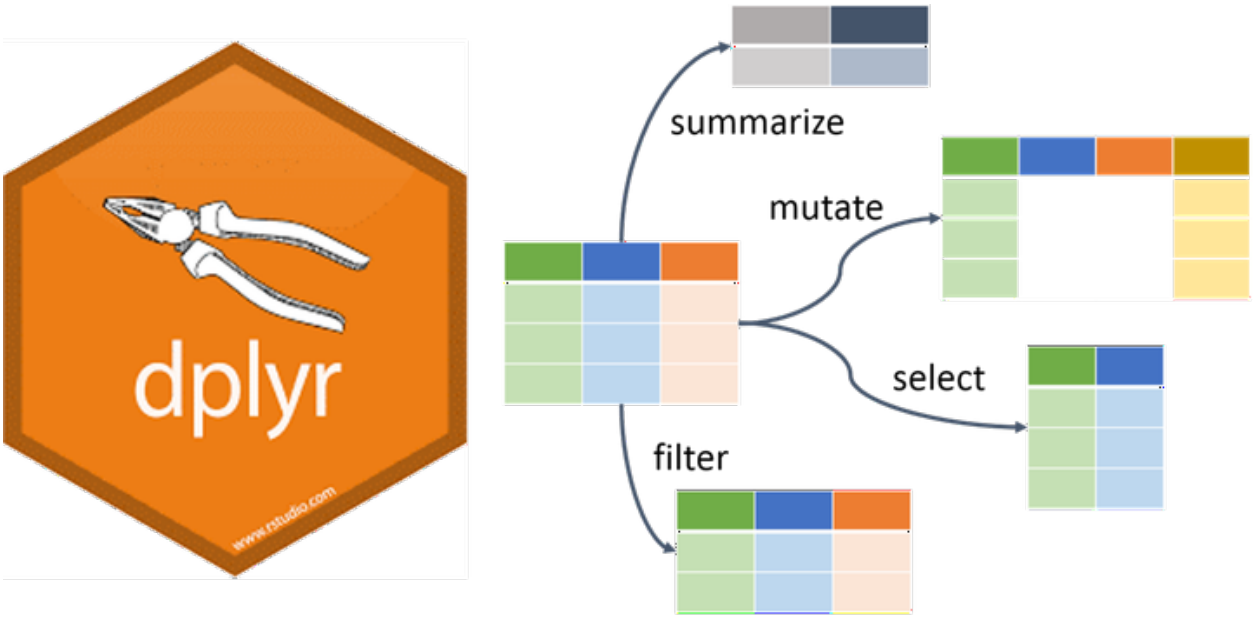
A nice shortcut is to supply the values you're interested in as a vector by typing `c(2, 4, 6, 8)`, where the `c` stands for concatenate which basically means to glue 2 to 4 to 6 to 8 all together in one vector.

Once we have that vector we can simply check if the number of hamsters for that row is `%in%` the vector we created. Here's how that looks:

```
hamsters |>
  filter(hamsters %in% c(2, 4, 6, 8))
```

```
## # A tibble: 2 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 América    female        6             3
## 2 Plascencia male          2             3
```

This can be a little bit confusing, so make sure you understand this. Think about the filter as happening row by row: first it checks the first row to see if 5 is in `c(2, 4, 6, 8)` – it isn't so it doesn't include that row. Then it checks the second row and also doesn't include that one because 7 isn't in the vector. It then checks the third row and keeps it because 6 is in there, and so on.



3.4 Filter with groups.

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>          <dbl>
## 1 Aleydis    female        5              2
## 2 Carolina   female        7              1
## 3 América    female        6              3
## 4 Plascencia male          2              3
## 5 Martín     male          1              4
```

```
hamsters |>
  group_by(gender) |>
  filter(max(hamster_cages) == 3)
```

```
## # A tibble: 3 x 4
## # Groups:   gender [1]
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>          <dbl>
## 1 Aleydis  female        5              2
## 2 Carolina female        7              1
## 3 América  female        6              3
```

We first grouped by gender. After that, every operation will happen at the group level instead of the row level.

The final line is where the magic happens. It tells R to return only the gender group where the max number of hamster cages is 3. This is the female group.

Notice that the male group isn't included because the max number of hamster cages is 4, not 3.

Similarly, we can get the gender group where the mean number of hamsters is 1.5. This time its the male group because there are two males – one with 1 hamster and the other with 2 hamsters.

```
hamsters |>
  group_by(gender) |>
  filter(mean(hamsters) == 1.5)
```

```
## # A tibble: 2 x 4
## # Groups:   gender [1]
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Plascencia male          2             3
## 2 Martín     male          1             4
```

The `n()` function is a shortcut for the number of rows in the group. So, the following code finds the gender group with 3 rows in it:

```
hamsters |>
  group_by(gender) |>
  filter(n() == 3)
```

```
## # A tibble: 3 x 4
## # Groups:   gender [1]
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Aleydis  female        5             2
## 2 Carolina female        7             1
## 3 América  female        6             3
```

Of course, there isn't any reason that we have to group by gender. We could instead group by the number of hamster cages.

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 Carolina   female        7             1
## 3 América    female        6             3
## 4 Plascencia male          2             3
## 5 Martín     male          1             4
```

```
hamsters |>
  group_by(hamster_cages) |>
  filter(n() == 1)
```

```
## # A tibble: 3 x 4
## # Groups:   hamster_cages [3]
##   name     gender hamsters hamster_cages
##   <chr>    <chr>     <dbl>         <dbl>
## 1 Aleydis  female        5             2
## 2 Carolina female        7             1
## 3 Martín   male          1             4
```

3.5 Mutate.

So far, `arrange()` has sorted our data, `select()` has chosen which columns to work with, and `filter()` has sorted which rows to use. We haven't changed our data at all yet though – that's what `mutate()` does.

For example, maybe we want to create a new variable based on the number of hamsters per cage for each person:

```
hamsters |>
  mutate(hamsters_per_cage = hamsters / hamster_cages)
```

```
## # A tibble: 5 x 5
##   name       gender hamsters hamster_cages hamsters_per_cage
##   <chr>      <chr>     <dbl>         <dbl>              <dbl>
## 1 Aleydis    female        5             2              2.5
```

```
## 2 Carolina   female        7             1             7
## 3 América    female        6             3             2
## 4 Plascencia male          2             3             0.667
## 5 Martín     male          1             4             0.25
```

Note that this is the first time we can start to combine the `{dplyr}` verbs to really make some magic happen.

```
hamsters |>
  mutate(hamsters_per_cage = hamsters / hamster_cages) |>
  arrange(hamsters_per_cage)
```

```
## # A tibble: 5 x 5
##   name       gender hamsters hamster_cages hamsters_per_cage
##   <chr>      <chr>     <dbl>         <dbl>             <dbl>
## 1 Martín     male          1             4             0.25
## 2 Plascencia male          2             3             0.667
## 3 América    female        6             3             2
## 4 Aleydis    female        5             2             2.5
## 5 Carolina   female        7             1             7
```

Or maybe we want an indicator of if the person has 5 or more hamsters:

```
hamsters |>
  mutate(five_or_more_hamsters = hamsters >= 5)
```

```
## # A tibble: 5 x 5
##   name       gender hamsters hamster_cages five_or_more_hamsters
##   <chr>      <chr>     <dbl>         <dbl> <lgl>
## 1 Aleydis    female        5             2 TRUE
## 2 Carolina   female        7             1 TRUE
## 3 América    female        6             3 TRUE
## 4 Plascencia male          2             3 FALSE
## 5 Martín     male          1             4 FALSE
```

We can also use `mutate()` to input new data:

```
hamsters |>
  mutate(cats = c(4, 5, 2, 3, 1))
```

```
## # A tibble: 5 x 5
```

```
##   name       gender hamsters hamster_cages  cats
##   <chr>      <chr>     <dbl>         <dbl> <dbl>
## 1 Aleydis    female        5             2     4
## 2 Carolina   female        7             1     5
## 3 América    female        6             3     2
## 4 Plascencia male          2             3     3
## 5 Martín     male          1             4     1
```

Of course, this only works if we give it the right amount of values:

Interestingly, we can give it just 1 value and it will repeat it the correct number of times automatically:

```
hamsters |>
  mutate(walruses = 0)
```

```
## # A tibble: 5 x 5
##   name       gender hamsters hamster_cages walruses
##   <chr>      <chr>     <dbl>         <dbl>    <dbl>
## 1 Aleydis    female        5             2        0
## 2 Carolina   female        7             1        0
## 3 América    female        6             3        0
## 4 Plascencia male          2             3        0
## 5 Martín     male          1             4        0
```

We can create multiple new columns with one use of mutate if we separate each new column with a ,:

```
hamsters |>
  mutate(hamsters_per_cage = round((hamsters / hamster_cages), 2),
         five_or_more_hamsters = hamsters >= 5)
```

```
## # A tibble: 5 x 6
##   name       gender hamsters hamster_cages hamsters_per_ca~ five_or_more_hamste~
##   <chr>      <chr>     <dbl>         <dbl>            <dbl> <lgl>
## 1 Aleydis    female        5             2             2.5  TRUE
## 2 Carolina   female        7             1             7    TRUE
## 3 América    female        6             3             2    TRUE
## 4 Plascencia male          2             3             0.67 FALSE
## 5 Martín     male          1             4             0.25 FALSE
```

Notice that `mutate()` leaves all of the original columns in the dataframe and adds new columns. If we instead use `transmute()` we'll only get the new columns:

```
hamsters |>
  transmute(hamsters_per_cage = hamsters / hamster_cages,
         five_or_more_hamsters = hamsters >= 5)
```

```
## # A tibble: 5 x 2
##   hamsters_per_cage five_or_more_hamsters
##               <dbl> <lgl>
## 1             2.5   TRUE
## 2             7     TRUE
## 3             2     TRUE
## 4             0.667 FALSE
## 5             0.25  FALSE
```

Slightly more complex things can be done by using values calculated from the data frame in the creation of a new column.

```
hamsters |>
  mutate(hamster_cages_centered = hamster_cages - mean(hamster_cages))
```

```
## # A tibble: 5 x 5
##   name       gender hamsters hamster_cages hamster_cages_centered
##   <chr>      <chr>     <dbl>         <dbl>                  <dbl>
## 1 Aleydis    female        5             2                   -0.6
## 2 Carolina   female        7             1                   -1.6
## 3 América    female        6             3                    0.4
## 4 Plascencia male          2             3                    0.4
## 5 Martín     male          1             4                    1.4
```

Notice that first the mean of the `hamster_cages` column is calculated to be 2.6, then the new column is created by subtracting 2.6 off of each value of the `hamster_cages` column.

3.6 Mutate with groups.

Sometimes its useful to define new variables based on a group. Remember groups tell R to operate on the data frame one group at a time as opposed to using all of the rows in the data frame.

For example, examine the following – note how it's different from the code above:

```
hamsters |>
  group_by(gender) |>
  mutate(hamster_cages_centered_by_gender = hamster_cages - mean(hamster_cages))
```

```
## # A tibble: 5 x 5
## # Groups:   gender [2]
##   name       gender hamsters hamster_cages hamster_cages_centered_by_gender
##   <chr>      <chr>     <dbl>         <dbl>                             <dbl>
## 1 Aleydis    female        5             2                               0
## 2 Carolina   female        7             1                              -1
## 3 América    female        6             3                               1
## 4 Plascencia male          2             3                              -0.5
## 5 Martín     male          1             4                               0.5
```

Before in `hamster_cages_centered` we subtracted the mean of `hamster_cages` which was 2.6 off of every value of `hamster_cages`.

Now, because we are grouping by gender, we subtract 2 off of `hamster_cages` for females and 3.5 off of `hamster_cages` for males. This is because `mean(hamster_cages)` operates on groups of rows defined by gender after we add the `group_by(gender)` attribute to the data frame.

3.7 Summarize.

`mutate()` kept the same number of rows in the data frame and added a column.

We also want to be able to collapse rows of a data frame which we might think of summarizing. One of the most common ways to summarize a set of numbers is to take the mean:

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 Carolina   female        7             1
## 3 América    female        6             3
## 4 Plascencia male          2             3
```

```
## 5 Martín     male          1             4
```

```
hamsters |>
  summarize(hamsters_mean = mean(hamsters))
```

```
## # A tibble: 1 x 1
##   hamsters_mean
##           <dbl>
## 1           4.2
```

Another common method of summarizing is the median. We can summarize multiple variables with multiple functions at the same time:

```
hamsters |>
  summarize(hamsters_mean = mean(hamsters),
            hamsters_median = median(hamsters),
            hamster_cages_mean = mean(hamster_cages),
            hamster_cages_median = median(hamster_cages))
```

```
## # A tibble: 1 x 4
##   hamsters_mean hamsters_median hamster_cages_mean hamster_cages_median
##           <dbl>           <dbl>              <dbl>                <dbl>
## 1           4.2               5                2.6                    3
```

3.8 Summarize with groups.

Summarize isn't that useful by itself, but when we add groups it becomes powerful.

It allows us to get a summary row for each group in the data frame:

```
hamsters
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Aleydis    female        5             2
## 2 Carolina   female        7             1
## 3 América    female        6             3
## 4 Plascencia male          2             3
## 5 Martín     male          1             4
```

```
hamsters |>
  group_by(gender) |>
  summarize(mean_hamsters = mean(hamsters))
```

```
## # A tibble: 2 x 2
##   gender mean_hamsters
##   <chr>          <dbl>
## 1 female           6
## 2 male             1.5
```

Just as before, we can create multiple summary statistics all at once:

```
hamsters |>
  group_by(gender) |>
  summarize(mean_hamsters = mean(hamsters),
            max_hamsters = max(hamsters),
            count = n())
```

```
## # A tibble: 2 x 4
##   gender mean_hamsters max_hamsters count
##   <chr>          <dbl>        <dbl> <int>
## 1 female           6              7     3
## 2 male             1.5            2     2
```

3.9 The power of combining verbs.

The true power of {dplyr} comes from combining these 5 verbs to solve problems. For example, see how we can piece commands together to do more and more complex operations:

```
hamsters |>
  arrange(hamsters)
```

```
## # A tibble: 5 x 4
##   name       gender hamsters hamster_cages
##   <chr>      <chr>     <dbl>         <dbl>
## 1 Martín     male          1             4
## 2 Plascencia male          2             3
## 3 Aleydis    female        5             2
## 4 América    female        6             3
```

```
## 5 Carolina   female        7             1
```

```
hamsters |>
  arrange(hamsters) |>
  select(-name)
```

```
## # A tibble: 5 x 3
##   gender hamsters hamster_cages
##   <chr>     <dbl>         <dbl>
## 1 male          1             4
## 2 male          2             3
## 3 female        5             2
## 4 female        6             3
## 5 female        7             1
```

```
hamsters |>
  arrange(hamsters) |>
  select(-name) |>
  mutate(walruses = 0)
```

```
## # A tibble: 5 x 4
##   gender hamsters hamster_cages walruses
##   <chr>     <dbl>         <dbl>    <dbl>
## 1 male          1             4        0
## 2 male          2             3        0
## 3 female        5             2        0
## 4 female        6             3        0
## 5 female        7             1        0
```

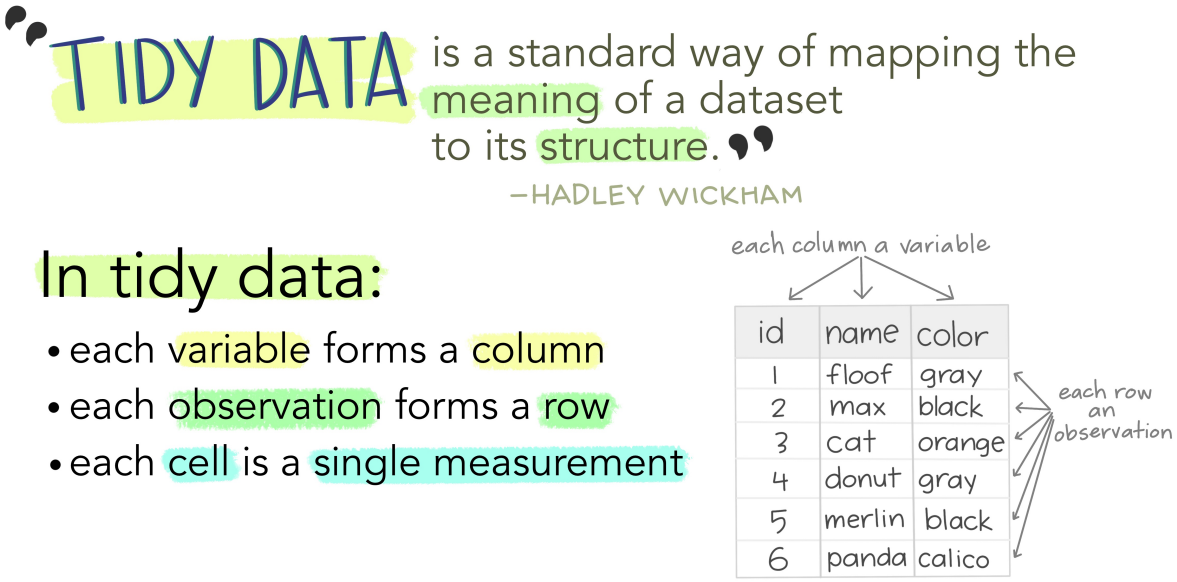
```
hamsters |>
  arrange(hamsters) |>
  select(-name) |>
  mutate(walruses = 0) |>
  group_by(gender) |>
  mutate(hamsters_centered_by_gender = hamsters - mean(hamsters))
```

```
## # A tibble: 5 x 5
## # Groups:   gender [2]
##   gender hamsters hamster_cages walruses hamsters_centered_by_gender
```

```
##   <chr>     <dbl>         <dbl>    <dbl>                        <dbl>
## 1 male          1             4        0                         -0.5
## 2 male          2             3        0                          0.5
## 3 female        5             2        0                         -1
## 4 female        6             3        0                          0
## 5 female        7             1        0                          1
```

```
hamsters |>
  arrange(hamsters) |>
  select(-name) |>
  mutate(walruses = 0) |>
  group_by(gender) |>
  mutate(hamsters_centered_by_gender = hamsters - mean(hamsters))
```

```
## # A tibble: 5 x 5
## # Groups:   gender [2]
##   gender hamsters hamster_cages walruses hamsters_centered_by_gender
##   <chr>     <dbl>         <dbl>    <dbl>                        <dbl>
## 1 male          1             4        0                         -0.5
## 2 male          2             3        0                          0.5
## 3 female        5             2        0                         -1
## 4 female        6             3        0                          0
## 5 female        7             1        0                          1
```



Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

4 Measuring the skill of fund managers.

Fidelity Contrafund (FCNTX) is a mutual fund operated and provided by Fidelity Investments. Its current manager is William Danoff, who has headed the fund since 1990. Contrafund's AUM (assets under management) as of July 2015 total over 112 billion USD. See https://www.fidelity.com/

4.1 Read the data.

Let's take the Kenneth R. French's Data Library to get the US market return:

https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

The `{assemble}` function.

```
# assemble makes the data ready to use.
assemble <- function(dat) {
  dat <- dat %>%
    # Rename only if there is a market factor.
    purrr::possibly(rename, otherwise = .)(Mkt_RF = `Mkt-RF`) %>%
    rename(date = X1) %>%
    # Date format requires this.
```

```
    slice(1:(which(is.na(date))[1] - 1)) %>%
    mutate(date = ymd(parse_date(as.character(date), format = "%Y%m"))) %>%
    mutate(date = date + months(1)) %>%
    mutate(date = rollback(date)) %>%
    # We work with decimal format, not percentage.
    mutate_if(sapply(., is.character), as.numeric) %>%
    mutate_at(vars(-date), function(x) x/100)
  return(dat)
}
```

Read the data.

```
f.ff <- read_csv("F-F_Research_Data_Factors.csv", skip = 3) |>
  assemble()
```

```
head(f.ff)
```

```
## # A tibble: 6 x 5
##   date        Mkt_RF     SMB     HML     RF
##   <date>       <dbl>   <dbl>   <dbl>  <dbl>
## 1 1926-07-31  0.0296 -0.023  -0.0287 0.0022
## 2 1926-08-31  0.0264 -0.014   0.0419 0.0025
## 3 1926-09-30  0.0036 -0.0132  0.0001 0.0023
## 4 1926-10-31 -0.0324  0.0004  0.0051 0.0032
## 5 1926-11-30  0.0253 -0.002  -0.0035 0.0031
## 6 1926-12-31  0.0262 -0.0004 -0.0002 0.0028
```

Functions to download the financial assets from the Internet and read them from the local directory. Here, we introduce `get_price_data()` and `get_return_data()` to get prices and returns for individual assets.

```
# Download price data.
get_price_data <- function(ticker, start, end) {
  price_data <- tq_get(ticker, from = start, to = end, get = 'stock.prices')
  return(price_data)
}

# Asset raw returns.
get_return_data <- function(closing_price, period = "monthly") {
```

```
  ret_data <- closing_price |>
    tq_transmute(select = adjusted, mutate_fun = periodReturn,
                 period = period, col_rename = 'returns')
  return(ret_data)
  }
```

Get the FCNTX asset data from Internet and join the factors.

```
# Download the price.
ticker <- c("FCNTX")
start <- "1980-01-01"
end <- "2020-12-31"
p.FCNTX <- get_price_data(ticker, start, end)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
r.FCNTX <- get_return_data(p.FCNTX)
```

4.2 Join databases.

```
# Join assets and factors: a.f
last_month_ff <- f.ff %>%
  select(date) %>%
  slice(nrow(f.ff)) %>%

  .[[1]]
a.f <- r.FCNTX |>
  filter(date <= last_month_ff) |>
  mutate(date = rollback(date, roll_to_first = TRUE)) |>
  mutate(date = date + months(1)) |>
  mutate(date = rollback(date)) |>
  left_join(f.ff,by = 'date') |>
  mutate(Re = returns - RF) |>
  relocate(Re, .before = RF)
```

Take a look of the resulting database.

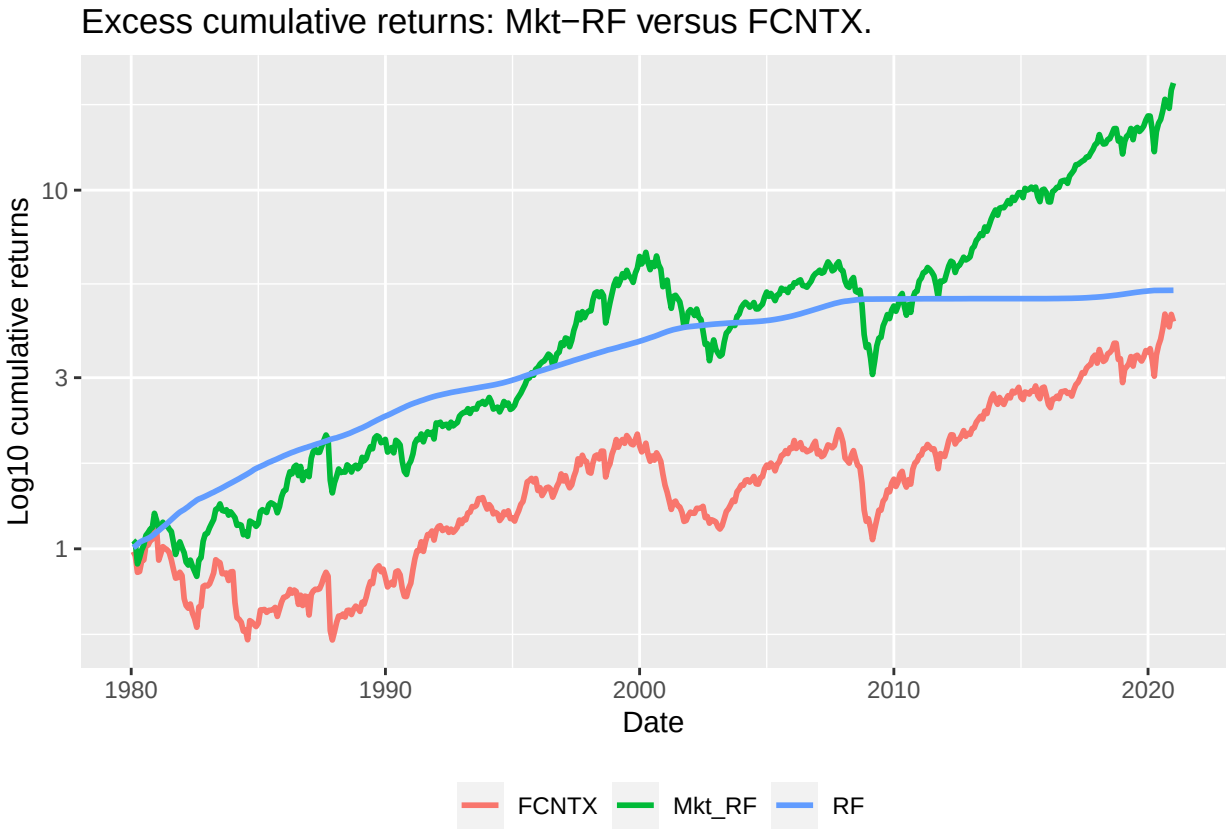
```
head(a.f)
```

```
## # A tibble: 6 x 7
##   date        returns  Mkt_RF     SMB     HML       Re     RF
##   <date>        <dbl>   <dbl>   <dbl>   <dbl>    <dbl>  <dbl>
## 1 1980-01-31 -0.00967  0.0551  0.0165  0.0172 -0.0177  0.008
## 2 1980-02-29 -0.0169  -0.0122 -0.0183  0.0066 -0.0258  0.0089
## 3 1980-03-31 -0.0894  -0.129  -0.0665 -0.0102 -0.102   0.0121
## 4 1980-04-30  0.0179   0.0397  0.0096  0.0101  0.00526 0.0126
## 5 1980-05-31  0.0789   0.0526  0.0216  0.0038  0.0708  0.0081
## 6 1980-06-30  0.0117   0.0306  0.0166 -0.0085  0.00564 0.0061
```

4.3 Cumulative returns.

```
a.f %>%
  rename(., !!ticker:=Re) |>
  select(date, Mkt_RF, ticker, RF) |>
  gather(Mkt_RF:RF, key = name, value = m.ret) |>
  group_by(name) |>
  mutate(c.ret = cumprod(1 + m.ret)) |>
  ggplot(aes(x = date, y = c.ret, color = name)) +
  geom_line(size = 1) +
  labs(y = "Log10 cumulative returns", x = "Date",
       title = "Excess cumulative returns: Mkt-RF versus FCNTX.") +
  theme(legend.position = "bottom", legend.title = element_blank()) +
  scale_y_log10()
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(ticker)` instead of `ticker` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```



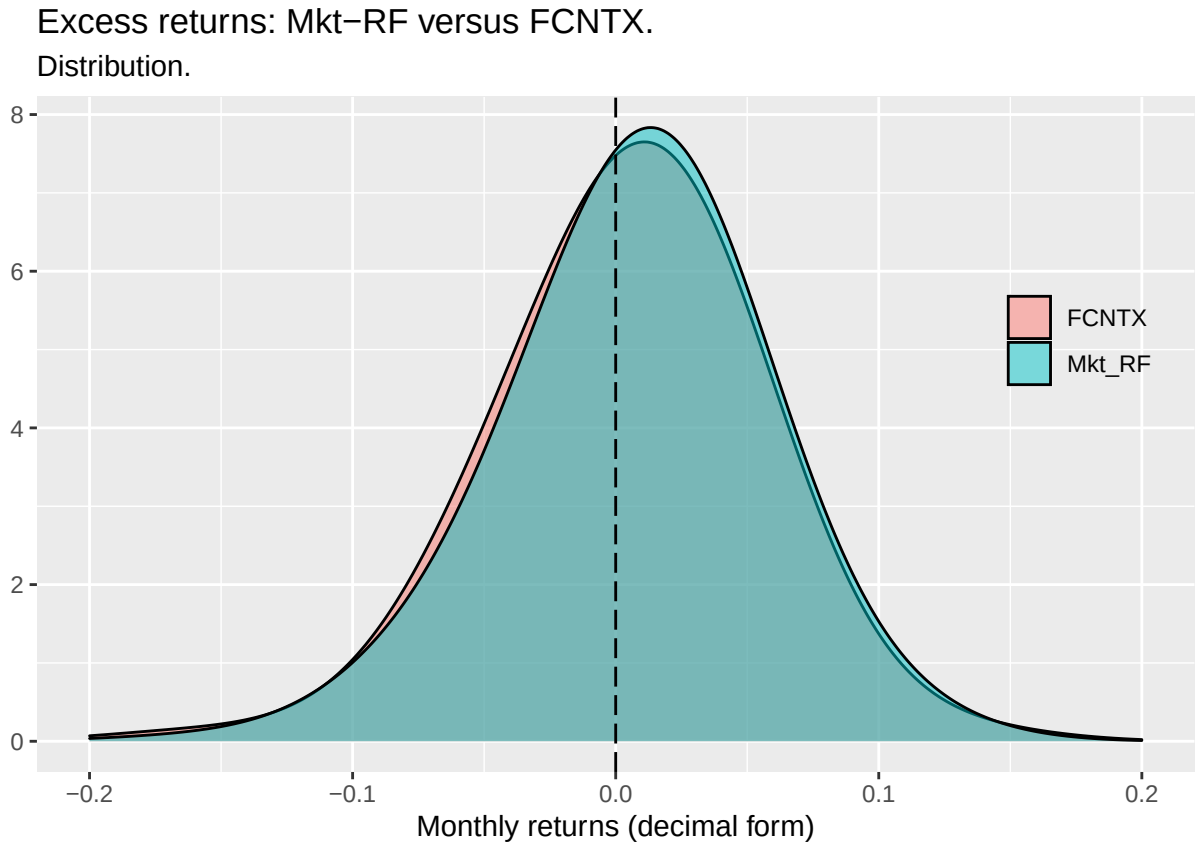
4.4 Distribution of returns.

The distribution of returns is not as informative as the cumulative returns.

```
a.f %>%
  rename(., !!ticker:=Re) |>
  gather(Mkt_RF, ticker, key = name, value = m.ret) |>
  group_by(name) |>
  ggplot(aes(m.ret, fill = name)) +
  geom_density(alpha = 0.5, adjust = 3) +
  labs(y = "", x = "Monthly returns (decimal form)",
       title = "Excess returns: Mkt-RF versus FCNTX.",
       subtitle = "Distribution.") +
  xlim(-0.2, 0.2) +
    theme(legend.title = element_blank(),
          legend.position = c(0.9, 0.65),
          legend.direction = "vertical",
          legend.background = element_blank()) +
```

```
  geom_vline(xintercept = 0, linetype = "longdash")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```



Even the risk-free asset is more attractive than a mutual fund.

5 25 Fama-French portfolios return visualization.

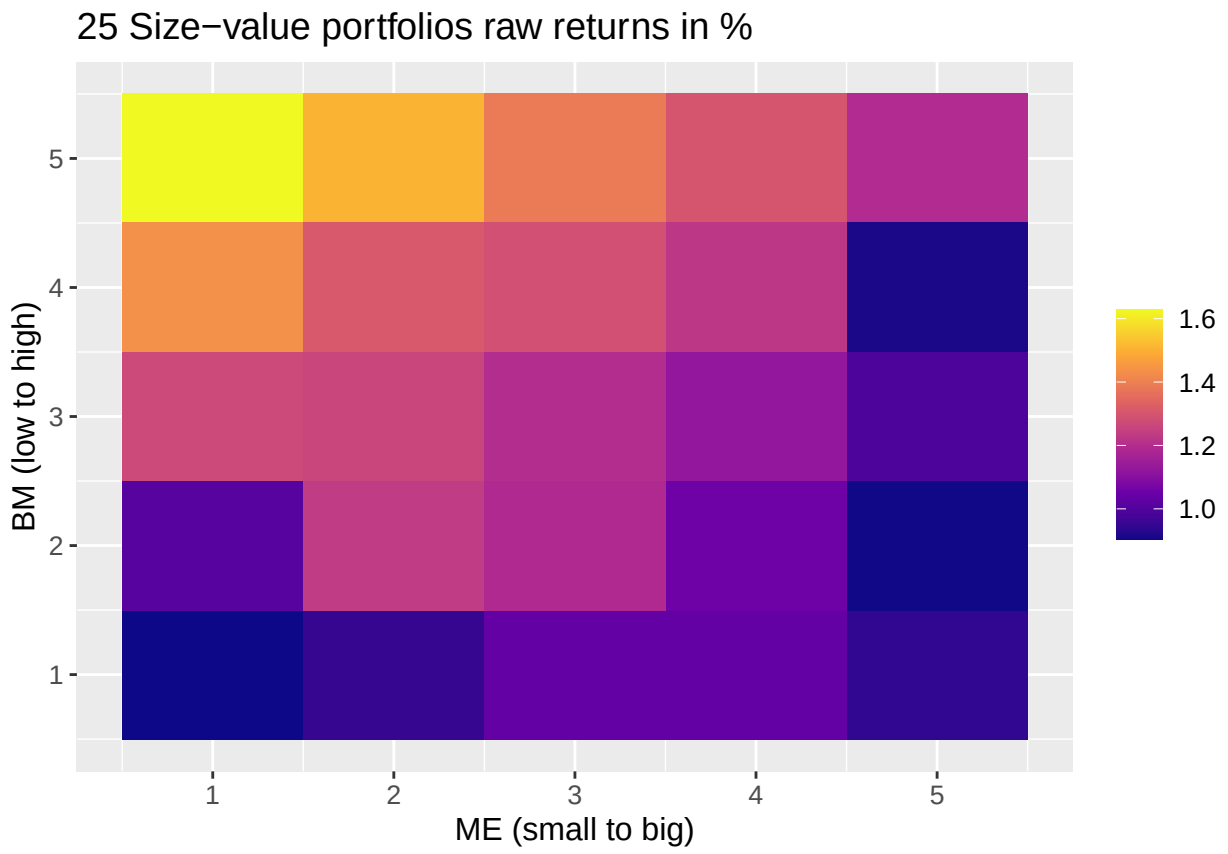
```
# Test portfolios: "25_Portfolios_5x5.csv"
p.25 <- read_csv("25_Portfolios_5x5.csv", skip = 15) %>%
  assemble()
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
```

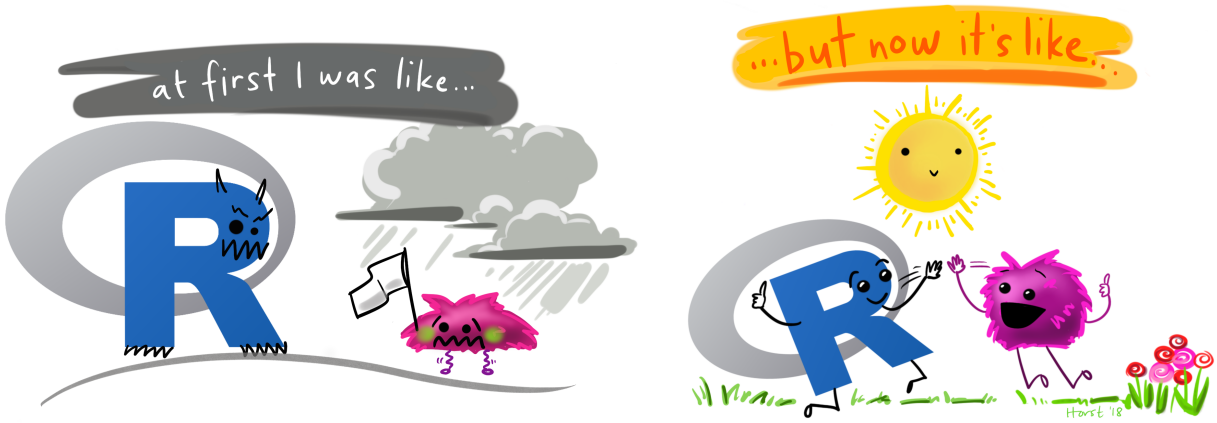
```
## )
## i Use `spec()` for the full column specifications.

## Warning: 273 parsing failures.
##  row        col   expected                                    actual
## 1136 X1         a double   Average Equal Weighted Returns -- Monthly '25_Portfolios_5
## 1136 NA         26 columns 1 columns                                   '25_Portfolios_5
## 1137 SMALL LoBM a double   SMALL LoBM                                  '25_Portfolios_5
## 1137 ME1 BM2    a double   ME1 BM2                                     '25_Portfolios_5
## 1137 ME1 BM3    a double   ME1 BM3                                     '25_Portfolios_5
## .... ......... ......... ..........................................  ................
## See problems(...) for more details.
```

```r
# Data.
r <- as.matrix(colMeans(p.25[2:26]), 5, 5)*100
ME <- c(rep(1, 5), rep(2, 5), rep(3, 5), rep(4, 5), rep(5, 5))
BM <- rep(1:5, 5)
temp.25 <- as.data.frame(cbind(r, ME, BM))
temp.25 <- as_tibble(temp.25)
# Plot.
fig.25 = ggplot(temp.25, aes(x = ME, y = BM)) +
  geom_raster(aes(fill = r)) +
  ggtitle("25 Size-value portfolios raw returns in %") +
          xlab("ME (small to big)") +
          ylab("BM (low to high)") +
  theme(legend.title = element_blank(), text = element_text(size = 12)) +
  scale_fill_viridis_c(option = "C")
fig.25
```




```
# Rayshader.
#plot_gg(fig.25, width = 4, height = 4, scale = 400, zoom = 0.7,
#        multicore = TRUE)
# Movie and picture.
#render_movie("p25.mp4", frames = 460)
#render_snapshot("p25.png", clear = TRUE)
```

5.1 Fact.



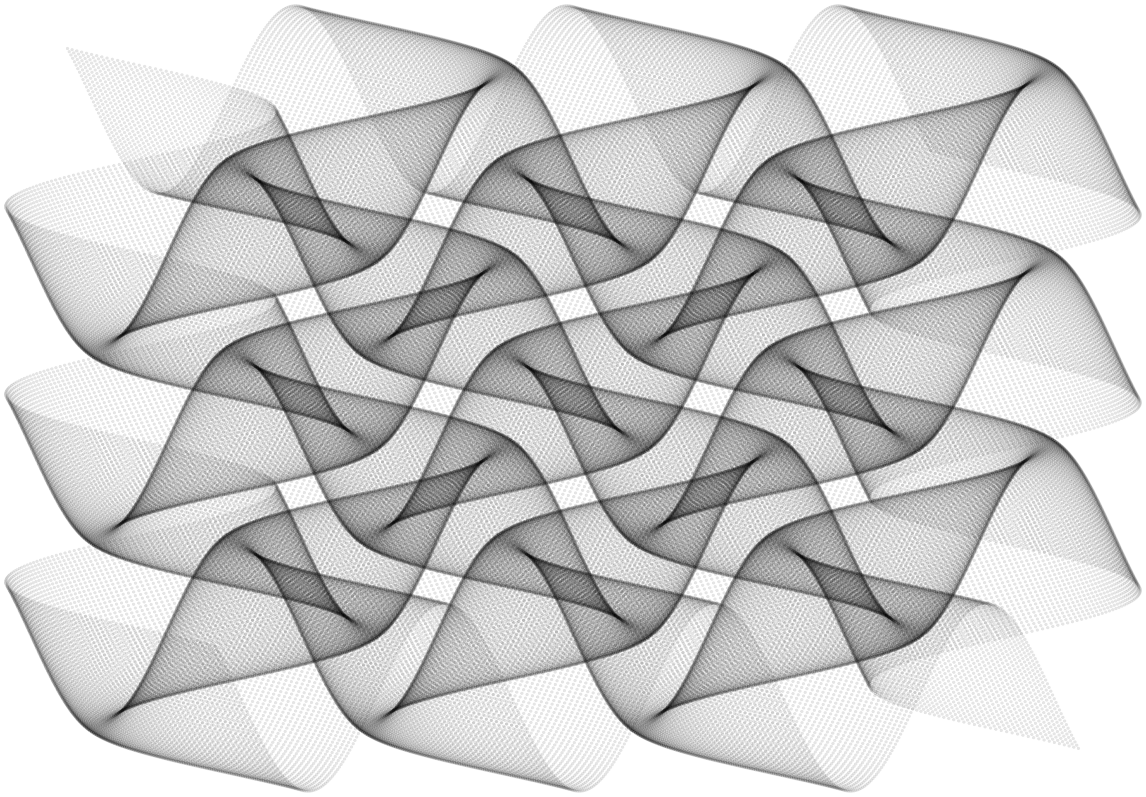
6 Art.

The following section is taken from Antonio Sánchez Chinchón. See:

- https://fronkonstin.com/
- https://www.hicetnunc.xyz/Antonio/creations

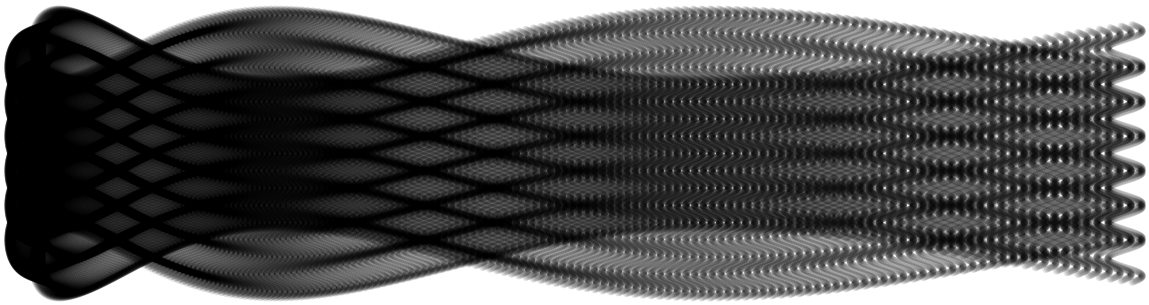
6.1 Silk.

```
seq(from = -10, to = 10, by = 0.04) %>%
  expand.grid(x = ., y=.) %>%
  ggplot(aes(x = (x + pi * sin(y)), y=(y + pi * sin(x)))) +
  geom_point(alpha = 0.1, shape = 20, size = 0) +
  theme_void()
```



6.2 A gel fish.

```
seq(from = -10, to = 10, by = 0.05) %>%
  expand.grid(x = ., y = .) %>%
  ggplot(aes(x = (x^2 + pi * cos(y)^2), y = (y + pi * sin(x)))) +
  geom_point(alpha = 0.1, shape = 20, size = 1, color = "black") +
  theme_void() +
  coord_fixed()
```



6.3 Abstract.

```
# This function creates the segments of the original polygon
polygon <- function(n) {
  tibble(
    x    = accumulate(1:(n - 1), ~.x + cos(.y * 2 * pi / n), .init = 0),
    y    = accumulate(1:(n - 1), ~.x + sin(.y * 2 * pi / n), .init = 0),
    xend = accumulate(2:n, ~.x + cos(.y * 2 * pi / n), .init = cos(2*pi/n)),
    yend = accumulate(2:n, ~.x + sin(.y * 2 * pi / n), .init = sin(2*pi/n)))
}

# This function creates segments from some mid-point of the edges
mid_points <- function(d, p, a, i, FUN = ratio_f) {
  d %>% mutate(
    angle=atan2(yend - y, xend - x) + a,
    radius = FUN(i),
    x = p * x + (1 - p) * xend,
```

```
    y = p * y + (1 - p) * yend,
    xend = x + radius * cos(angle),
    yend = y + radius * sin(angle)) %>%
    select(x, y, xend, yend)
}

# This function connect the ending points of mid-segments
con_points <- function(d) {
  d %>% mutate(
    x = xend,
    y = yend,
    xend = lead(x, default = first(x)),
    yend = lead(y, default = first(y))) %>%
    select(x, y, xend, yend)
}

edges <- 3   # Number of edges of the original polygon
niter <- 250 # Number of iterations
pond <- 0.24  # Weight to calculate the point on the middle of each edge
step  <- 13  # Number of times to draw mid-segments before connect ending points
alph  <- 0.25 # transparency of curves in geom_curve
angle <- 0.6 # angle of mid-segment with the edge
curv <- 0.1   # Curvature of curves
line_color <- "purple" # Color of curves in geom_curve
back_color <- "pink" # Background of the ggplot
ratio_f <- function(x) {sin(x)} # To calculate the longitude of mid-segments

# Generation on the fly of the dataset
accumulate(.f = function(old, y) {
  if (y%%step!=0) mid_points(old, pond, angle, y) else con_points(old)
  }, 1:niter,
.init=polygon(edges)) %>% bind_rows() -> df

# Plot
ggplot(df) +
  geom_curve(aes(x = x, y = y, xend = xend, yend = yend),
```

```
            curvature = curv,
            color = line_color,
            alpha = alph) +
  coord_equal() +
  theme(legend.position  = "none",
        panel.background = element_rect(fill=back_color),
        plot.background  = element_rect(fill=back_color),
        axis.ticks       = element_blank(),
        panel.grid       = element_blank(),
        axis.title       = element_blank(),
        axis.text        = element_blank())
```

6.4 Self portrait.

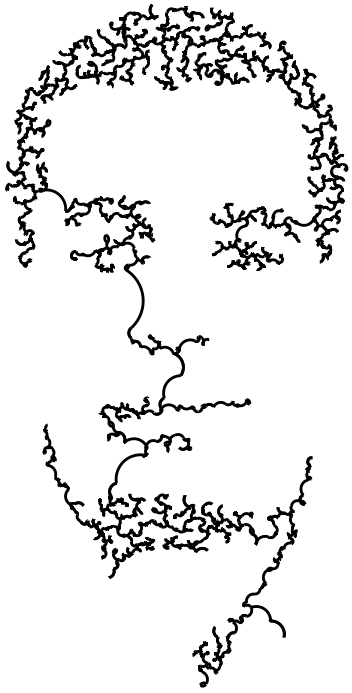
```
# Location of the photograph
file="ml.png"
# Load, convert to grayscale, filter image (to convert it to bw) and sample
```

```
load.image(file) %>%
  grayscale() %>%
  threshold("10%") %>% #45%
  as.cimg() %>%
  as.data.frame() -> franky

clustResultx <- function(x) {
  clustCut <- tibble(cluster_id = cutree(clusters, x)) %>% bind_cols(data)
  clustCut %>% group_by(cluster_id) %>%
    summarize(size = n()) %>%
    filter(size > 1) %>%
    select(cluster_id) %>%
    inner_join(clustCut, by = "cluster_id") -> clustCut
  return(clustCut)
  }

add_segments <- function(x){

  df1 <- clustEvol[[x]]
  df0 <- clustEvol[[x-1]]

  new_points <- anti_join(df1, df0, by = "id")

  new_points %>%
    inner_join(df1, by = "cluster_id", suffix = c(".1", ".2")) %>%
    filter(id.1 != id.2) %>%
    mutate(d = sqrt((x.1 - x.2)^2 + (y.1 - y.2)^2)) %>%
    group_by(id.1) %>%
    arrange(d) %>%
    slice(1) %>%
    select(p1 = id.1, p2 = id.2) %>%
    ungroup -> new_segments1

  new_points %>% anti_join(bind_rows(select(new_segments1, id = p1),
                                     select(new_segments1, id = p2)),
                           by = "id") %>%
```

```
    group_by(cluster_id) %>%
    ungroup -> unpaired_points

  unpaired_points %>% inner_join(unpaired_points, by = "cluster_id",
                                 suffix = c(".1", ".2")) %>%
    filter(id.1 < id.2) %>%
    select(p1 = id.1, p2 = id.2) -> new_segments2

  new_points <- anti_join(df1, df0, by = c("id", "cluster_id"))

  new_points %>%
    inner_join(df1, by = "cluster_id", suffix = c(".1", ".2")) %>%
    filter(id.1 != id.2) %>%
    anti_join(new_points, by = c("id.2" = "id")) %>%
    mutate(d = sqrt((x.1 - x.2)^2 + (y.1 - y.2)^2)) %>%
    arrange(d) %>%
    slice(1) %>%
    select(p1 = id.1, p2 = id.2) %>%
    ungroup -> new_segments3

  bind_rows(new_segments1, new_segments2, new_segments3)
}

# Sample size
n <- 1500 # 2500

set.seed(1)
franky %>%
  sample_n(n, weight=(1-value)) %>%
  select(x,y) %>% mutate(id = row_number()) -> data

dist_data <- dist(data %>% select(-id), method = "euclidean")

# Hierarchical clustering
clusters <- hclust(dist_data, method = 'single')
```

```r
nrow(data):1 %>%
  map(function(x) clustResultx(x)) -> clustEvol

2:length(clustEvol) %>%
  map(function(x) add_segments(x)) %>%
  bind_rows() -> segments_id

segments_id %>%
  inner_join(data, by = c("p1" = "id"), suffix = c(".1", ".2")) %>%
  inner_join(data, by = c("p2" = "id"), suffix = c(".1", ".2")) %>%
  select(x = x.1, y = y.1, xend = x.2, yend = y.2) -> segments

# Plot
ggplot(segments) +
  geom_curve(aes(x = x, y = y, xend = xend, yend = yend),
             ncp = 20) +
  scale_x_continuous(expand=c(.1, .1)) +
  scale_y_continuous(expand=c(.1, .1), trans = scales::reverse_trans()) +
    labs(title = "Reinterpretation of myself.") +
  coord_equal() +
  theme_void()
```

Reinterpretation of myself.



7 Maps.

The following section is taken from Diego Valle. See https://www.diegovalle.net/

Install `{mxmaps}`.

```
head(df_mxmunicipio_2020)
```

```
## # A tibble: 6 x 17
##   state_code municipio_code region state_name     state_name_official state_abbr
##   <chr>      <chr>          <chr>  <chr>          <chr>               <chr>
## 1 01         001            01001  Aguascalientes Aguascalientes      AGS
## 2 01         002            01002  Aguascalientes Aguascalientes      AGS
## 3 01         003            01003  Aguascalientes Aguascalientes      AGS
## 4 01         004            01004  Aguascalientes Aguascalientes      AGS
## 5 01         005            01005  Aguascalientes Aguascalientes      AGS
## 6 01         006            01006  Aguascalientes Aguascalientes      AGS
## # ... with 11 more variables: state_abbr_official <chr>, municipio_name <chr>,
```

```
## #   year <dbl>, pop <int>, pop_male <int>, pop_female <int>, afromexican <int>,
## #   indigenous_language <int>, metro_area <chr>, long <dbl>, lat <dbl>
```

7.1 Indigenous language.

Percentage of Mexican population that speaks an indigenous language by municipality.

```
df_mxmunicipio_2020$value <-  (df_mxmunicipio_2020$indigenous_language /
                               df_mxmunicipio_2020$pop)*100
```

The last column `value` in `df_mxmunicipio_2020` will change according to our examples. See the new variable `value`:

```
head(df_mxmunicipio_2020)
```

```
## # A tibble: 6 x 18
##   state_code municipio_code region state_name     state_name_official state_abbr
##   <chr>      <chr>          <chr>  <chr>          <chr>               <chr>
## 1 01         001            01001  Aguascalientes Aguascalientes      AGS
## 2 01         002            01002  Aguascalientes Aguascalientes      AGS
## 3 01         003            01003  Aguascalientes Aguascalientes      AGS
## 4 01         004            01004  Aguascalientes Aguascalientes      AGS
## 5 01         005            01005  Aguascalientes Aguascalientes      AGS
## 6 01         006            01006  Aguascalientes Aguascalientes      AGS
## # ... with 12 more variables: state_abbr_official <chr>, municipio_name <chr>,
## #   year <dbl>, pop <int>, pop_male <int>, pop_female <int>, afromexican <int>,
## #   indigenous_language <int>, metro_area <chr>, long <dbl>, lat <dbl>,
## #   value <dbl>
```

You may want to see it clearer:

```
df_mxmunicipio_2020 |>
  select(municipio_name, pop, indigenous_language, value)
```

```
## # A tibble: 2,469 x 4
##    municipio_name         pop indigenous_language  value
##    <chr>                <int>               <int>  <dbl>
##  1 Aguascalientes      948990                1839 0.194
##  2 Asientos             51536                  22 0.0427
##  3 Calvillo             58250                  76 0.130
```

```
##  4 Cosío                17000                   7 0.0412
##  5 Jesús María         129929                 158 0.122
##  6 Pabellón de Arteaga  47646                  52 0.109
##  7 Rincón de Romos      57369                 213 0.371
##  8 San José de Gracia    9552                  21 0.220
##  9 Tepezalá             22485                  18 0.0801
## 10 El Llano             20853                  13 0.0623
## # ... with 2,459 more rows
```

Even clearer because we are interested in the highest values in `value`:

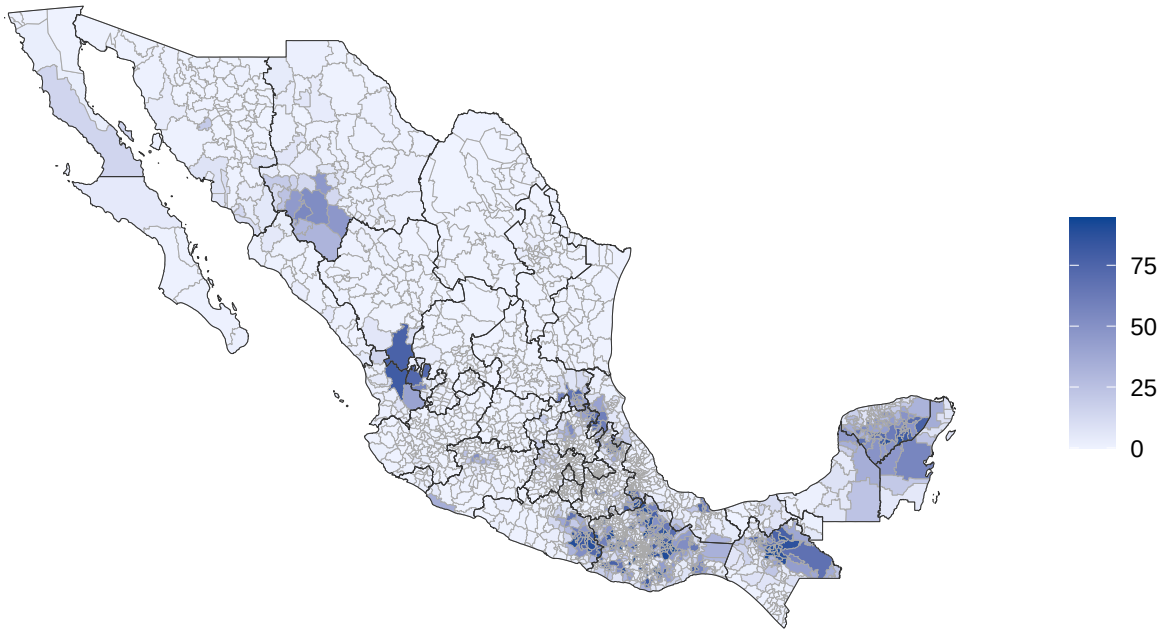
```
df_mxmunicipio_2020 |>
  select(state_abbr, municipio_name, pop, indigenous_language, value) |>
  arrange(desc(value))
```

```
## # A tibble: 2,469 x 5
##    state_abbr municipio_name            pop indigenous_language value
##    <chr>      <chr>                   <int>               <int> <dbl>
##  1 OAX        Santa Catalina Quierí     825                 778  94.3
##  2 OAX        San Juan Yatzona          440                 410  93.2
##  3 OAX        Santa Inés Yatzeche       908                 844  93.0
##  4 OAX        San Miguel Quetzaltepec  7286                6768  92.9
##  5 OAX        Mixistlán de la Reforma  2487                2306  92.7
##  6 OAX        San Miguel Yotao          585                 541  92.5
##  7 OAX        San Pedro Yaneri          867                 801  92.4
##  8 OAX        San José Lachiguiri      3700                3418  92.4
##  9 OAX        San Juan Yaeé            1426                1317  92.4
## 10 OAX        San Pedro Ocotepec       2404                2217  92.2
## # ... with 2,459 more rows
```

Oaxaca and Chiapas in the highest levels. Let's plot.

```
gg <- MXMunicipioChoropleth$new(df_mxmunicipio_2020)
gg$title <- "Percentage of the population that speaks an indigenous language."
gg$set_num_colors(1)
gg$render()
```

Percentage of the population that speaks an indigenous language.

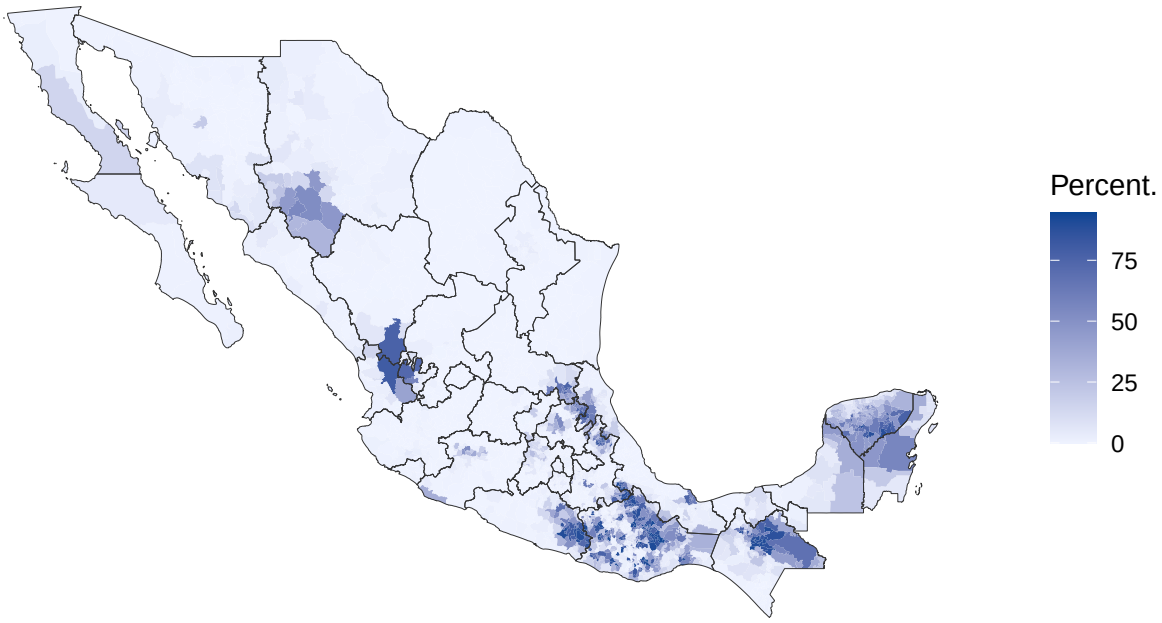


Lines that define municipalities are problematic to see a clear picture. Let's drop them or at least make them transparent.

Plot again:

```
p <- mxmunicipio_choropleth(df_mxmunicipio_2020,
    title = "Percentage of the population that speaks an indigenous language.",
    legend = "Percent.", num_colors = 1)
p[["layers"]][[1]][["aes_params"]][["colour"]] <- "transparent"
p
```

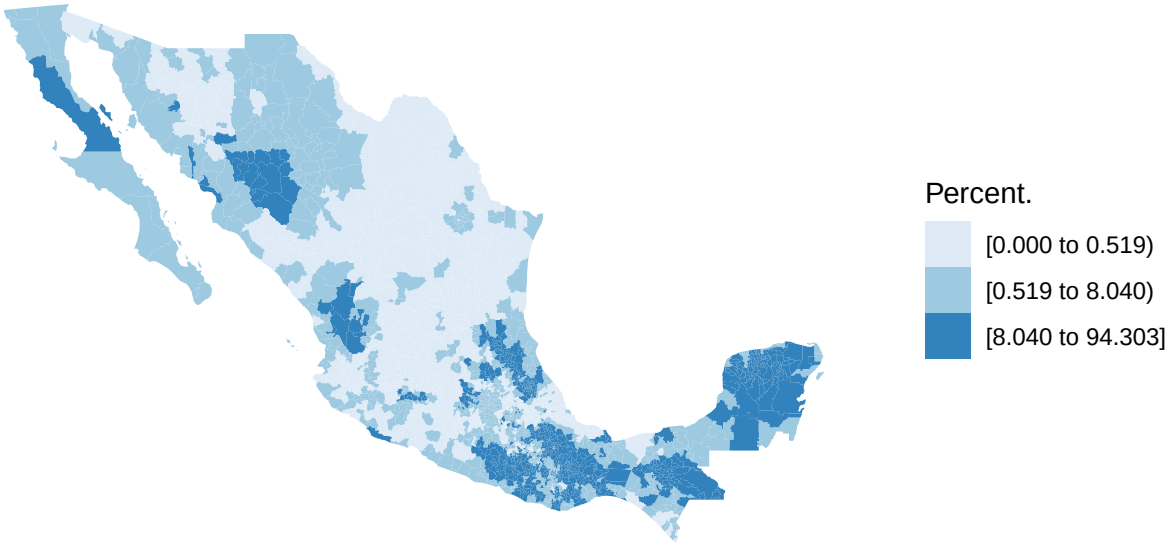
Percentage of the population that speaks an indigenous language.



We can also hide the states.

```
p <- mxmunicipio_choropleth(df_mxmunicipio_2020,
    title = "Percentage of the population that speaks an indigenous language.",
    legend = "Percent.", num_colors = 3, show_states = "FALSE")
p[["layers"]][[1]][["aes_params"]][["colour"]] <- "transparent"
p
```

Percentage of the population that speaks an indigenous language.



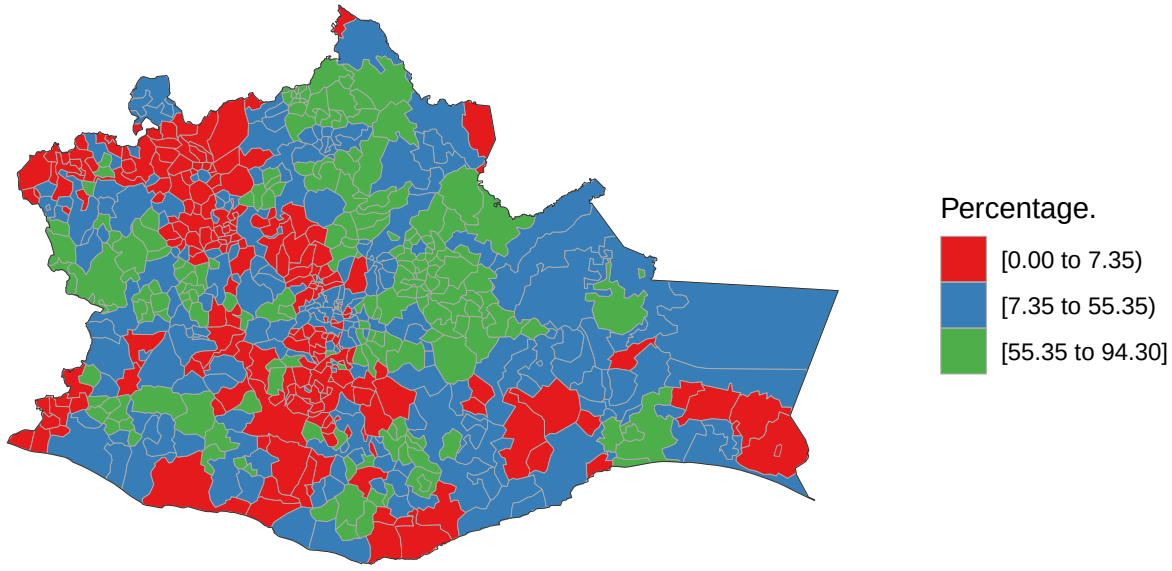
Nice. Let's zoom for a single state.

```
df_mxmunicipio_2020 |>
  select(state_abbr, municipio_name, pop, indigenous_language, value) |>
  filter(state_abbr == "NL") |>
  arrange(desc(value))
```

```
## # A tibble: 51 x 5
##    state_abbr municipio_name            pop indigenous_language value
##    <chr>      <chr>                   <int>               <int> <dbl>
##  1 NL         Pesquería              147624                6589  4.46
##  2 NL         El Carmen              104478                4181  4.00
##  3 NL         García                 397205               13235  3.33
##  4 NL         Marín                    5119                 168  3.28
##  5 NL         General Zuazua         102149                3338  3.27
##  6 NL         Ciénega de Flores       68747                2146  3.12
##  7 NL         Salinas Victoria        86766                2541  2.93
##  8 NL         San Pedro Garza García 132169                2523  1.91
```

```
##  9 NL          Juárez                  471523                 8798  1.87
## 10 NL          Santiago                 46784                  782  1.67
## # ... with 41 more rows
```

```
gg = MXMunicipioChoropleth$new(df_mxmunicipio_2020)
gg$title <- "Population that speaks an indigenous language."
gg$set_num_colors(3)
gg$set_zoom(subset(df_mxmunicipio_2020, state_name %in% c("Oaxaca"))$region)
gg$ggplot_scale <- scale_fill_brewer("Percentage.", type = "qual", palette = 6,
                                     na.value = "gray")
p <- gg$render()
p + theme_void()
```

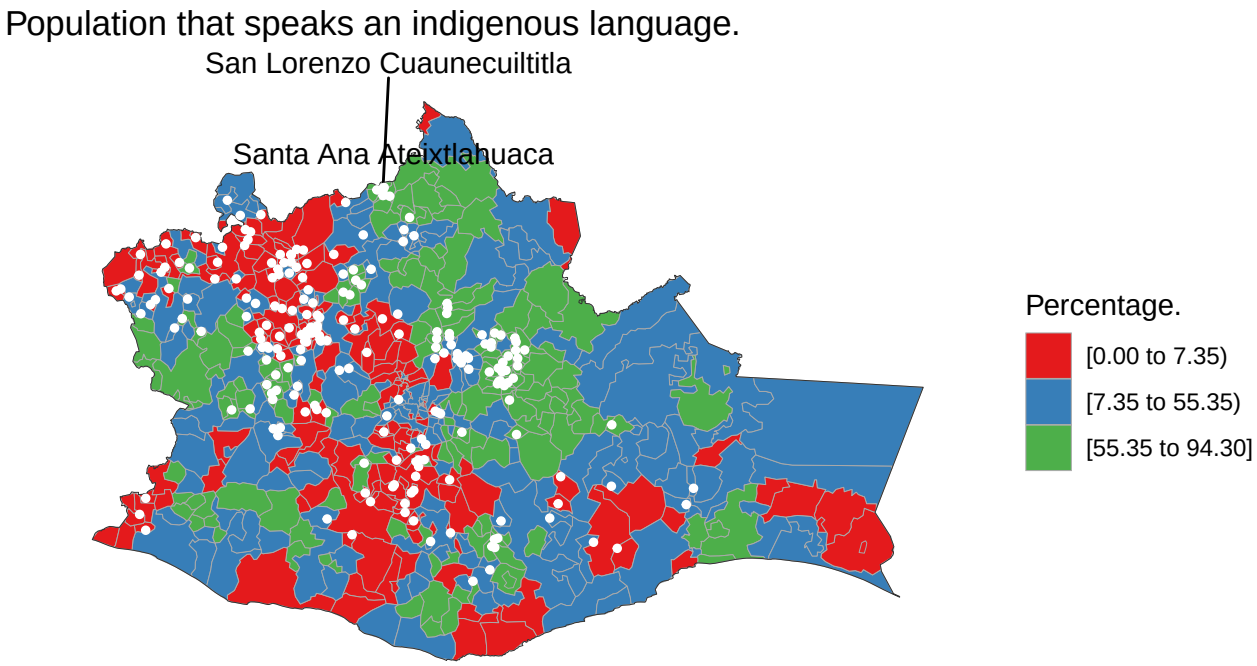


Not sure where are the smaller cities?

```
st <- subset(df_mxmunicipio_2020, state_name %in% c("Oaxaca"))
labels <- st
labels$group <- NA
labels <- subset(labels, pop < 2e03)
```

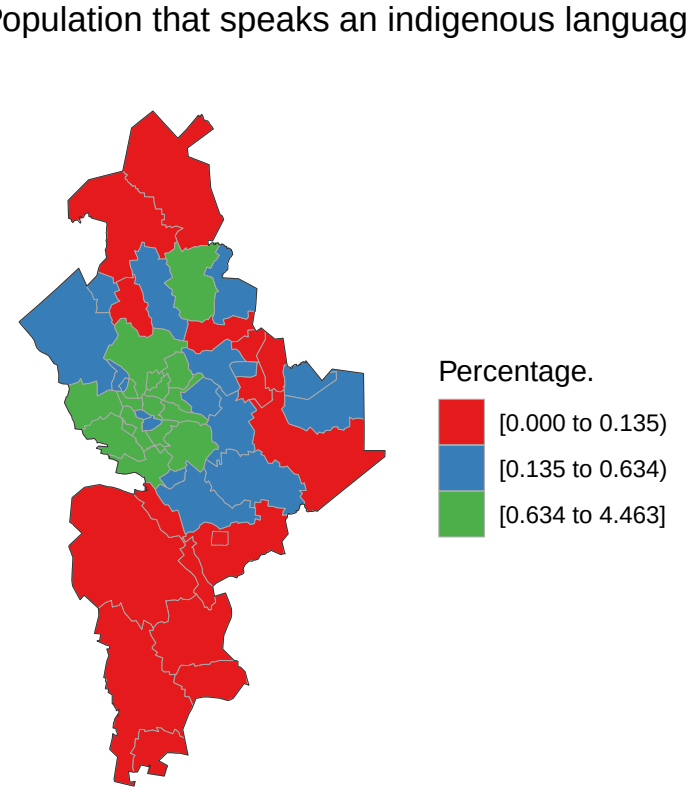
```
p + geom_text_repel(data = labels, aes(long, lat, label = municipio_name),
                    nudge_x = 0.1, nudge_y = 0.7) +
  geom_point(data = labels, aes(long, lat), color = "white", size = 1)
```

```
## Warning: ggrepel: 223 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



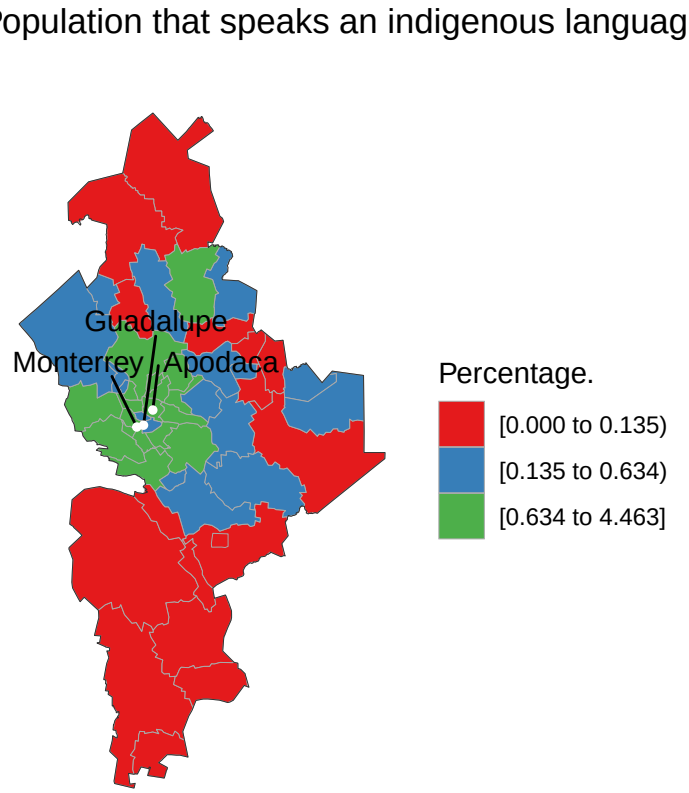
Let's see Nuevo León.

```
gg = MXMunicipioChoropleth$new(df_mxmunicipio_2020)
gg$title <- "Population that speaks an indigenous language."
gg$set_num_colors(3)
gg$set_zoom(subset(df_mxmunicipio_2020, state_name %in% c("Nuevo León"))$region)
gg$ggplot_scale <- scale_fill_brewer("Percentage.", type = "qual", palette = 6,
                                     na.value = "gray")
p <- gg$render()
p + theme_void()
```



Not sure where are the main cities?

```
st <- subset(df_mxmunicipio_2020, state_name %in% c("Nuevo León"))
labels <- st
labels$group <- NA
labels <- subset(labels, pop > 5e05)

p + geom_text_repel(data = labels, aes(long, lat, label = municipio_name),
                    nudge_x = 0.1, nudge_y = 0.7) +
  geom_point(data = labels, aes(long, lat), color = "white", size = 1)
```



7.2 Gender.

Let's explor female population by municipality. Note that we re-define `value`.

```
df_mxmunicipio_2020$value <-
        (df_mxmunicipio_2020$pop_female / df_mxmunicipio_2020$pop)*100
```

See the results.

```
head(df_mxmunicipio_2020)
```

```
## # A tibble: 6 x 18
##   state_code municipio_code region state_name     state_name_official state_abbr
##   <chr>      <chr>          <chr>  <chr>          <chr>               <chr>
## 1 01         001            01001  Aguascalientes Aguascalientes      AGS
## 2 01         002            01002  Aguascalientes Aguascalientes      AGS
## 3 01         003            01003  Aguascalientes Aguascalientes      AGS
## 4 01         004            01004  Aguascalientes Aguascalientes      AGS
## 5 01         005            01005  Aguascalientes Aguascalientes      AGS
```

```
## 6 01         006            01006  Aguascalientes Aguascalientes      AGS
## # ... with 12 more variables: state_abbr_official <chr>, municipio_name <chr>,
## #   year <dbl>, pop <int>, pop_male <int>, pop_female <int>, afromexican <int>,
## #   indigenous_language <int>, metro_area <chr>, long <dbl>, lat <dbl>,
## #   value <dbl>
```

That was a little bit hard to see. Let's select a few columns and arrange `value`:

```
df_mxmunicipio_2020 |>
  select(state_abbr, municipio_name, pop_female, value) |>
  arrange(desc(value))
```

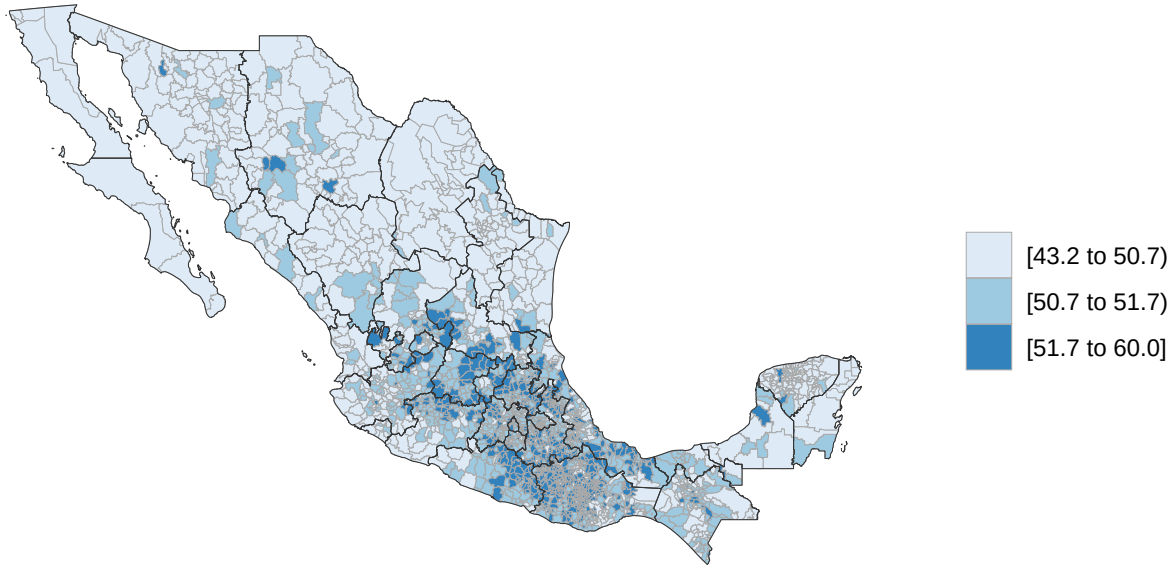
```
## # A tibble: 2,469 x 4
##    state_abbr municipio_name            pop_female value
##    <chr>      <chr>                          <int> <dbl>
##  1 OAX        Santiago Tepetlapa                78  60
##  2 OAX        San Juan Mixtepec.               362  59.6
##  3 OAX        San Bartolomé Quialana          1395  58.4
##  4 OAX        San Lucas Quiaviní               992  57.7
##  5 OAX        Abejones                         481  57.2
##  6 PUE        Teteles de Ávila Castillo       3766  56.6
##  7 OAX        Santa Catarina Ticuá             558  56.6
##  8 OAX        San Miguel Tulancingo            173  56.4
##  9 OAX        Santa María la Asunción         1827  56.1
## 10 OAX        Santa Ana Yareni                 356  55.9
## # ... with 2,459 more rows
```

Oaxaca again?

Let's plot.

```
gg <- MXMunicipioChoropleth$new(df_mxmunicipio_2020)
gg$title <- "Percentage of female population."
gg$set_num_colors(3)
gg$render()
```

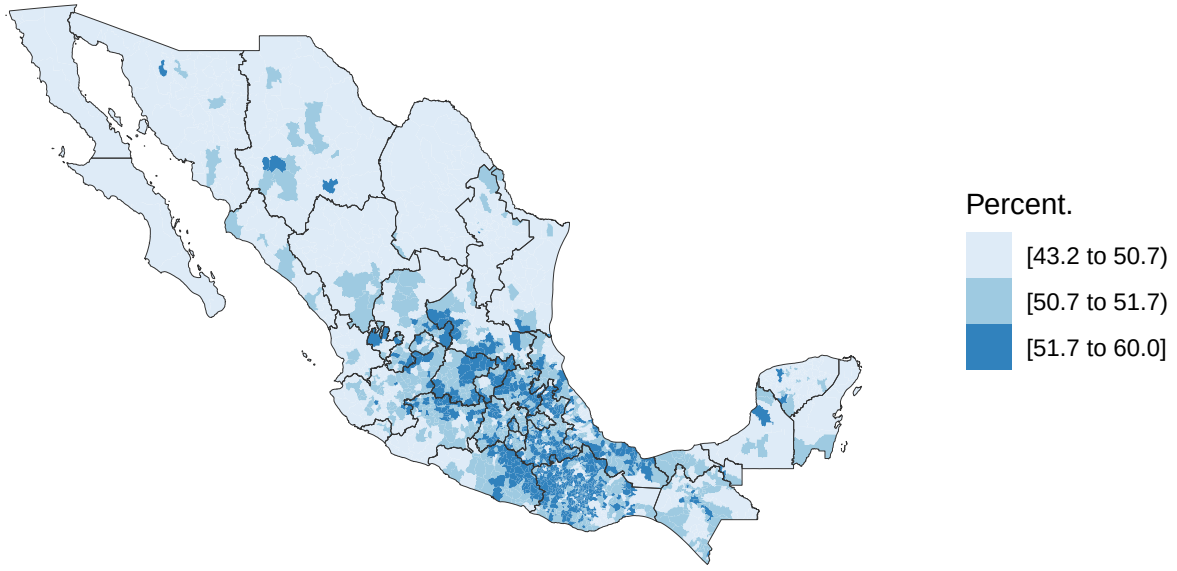
Percentage of female population.



Those border lines to identify municipalities are annoying. Plot again:

```
p <- mxmunicipio_choropleth(df_mxmunicipio_2020,
    title = "Percentage of female population.",
   legend = "Percent.", num_colors = 3)
p[["layers"]][[1]][["aes_params"]][["colour"]] <- "transparent"
p
```

Percentage of female population.

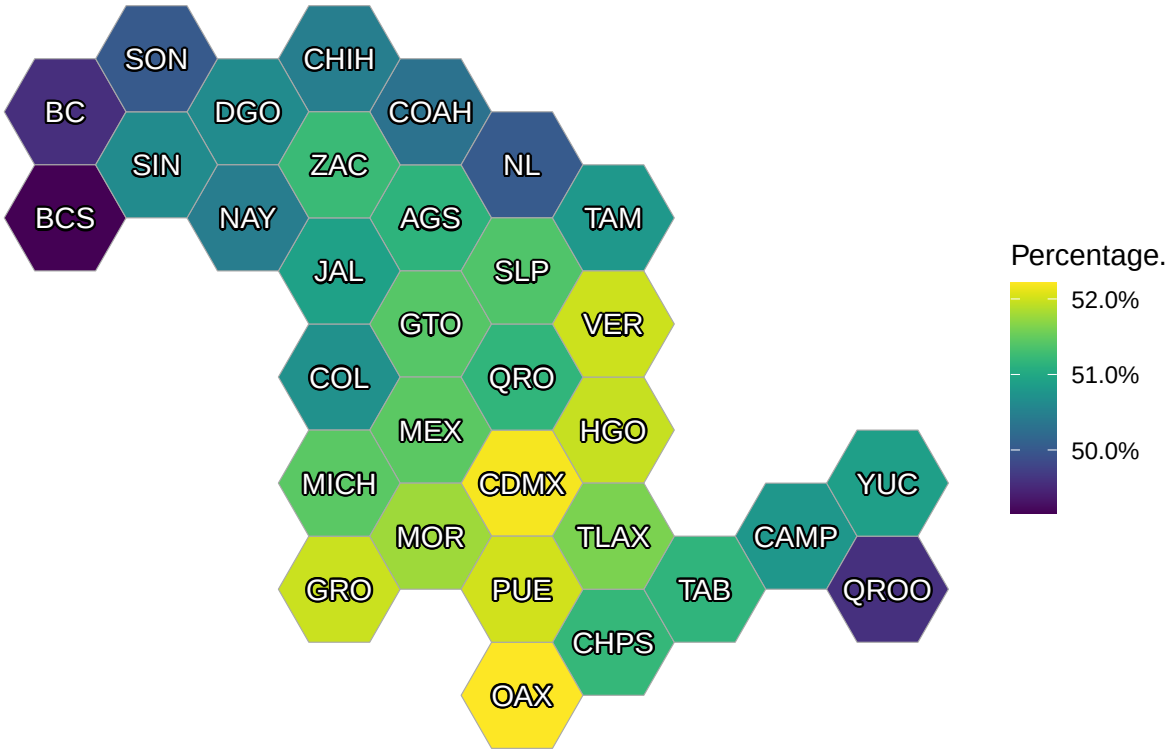


The term hexbin map refers to an unusual geospatial object where all regions of the map are represented as hexagons.

```
df_mxstate_2020$value = df_mxstate_2020$pop_female / df_mxstate_2020$pop
mxhexbin_choropleth(df_mxstate_2020,
                    num_colors = 1,
                    label_color = "white",
                    shadow_color = "black",
                    title = "Percentage of female population.",
                    legend = "%",
                    label_size = 3.8) +
  scale_fill_viridis("Percentage.", labels = percent)
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

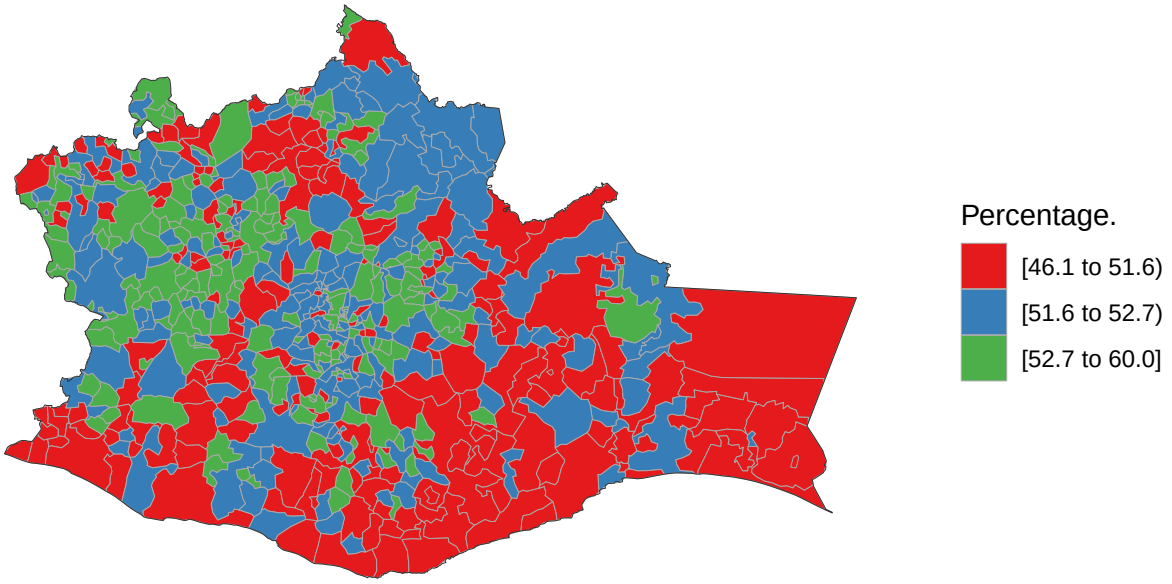
Percentage of female population.



Nice.

```
gg = MXMunicipioChoropleth$new(df_mxmunicipio_2020)
gg$title <- "Percentage of female population."
gg$set_num_colors(3)
gg$set_zoom(subset(df_mxmunicipio_2020, state_name %in% c("Oaxaca"))$region)
gg$ggplot_scale <- scale_fill_brewer("Percentage.", type = "qual", palette = 6,
                                     na.value = "gray")
p <- gg$render()
p + theme_void()
```

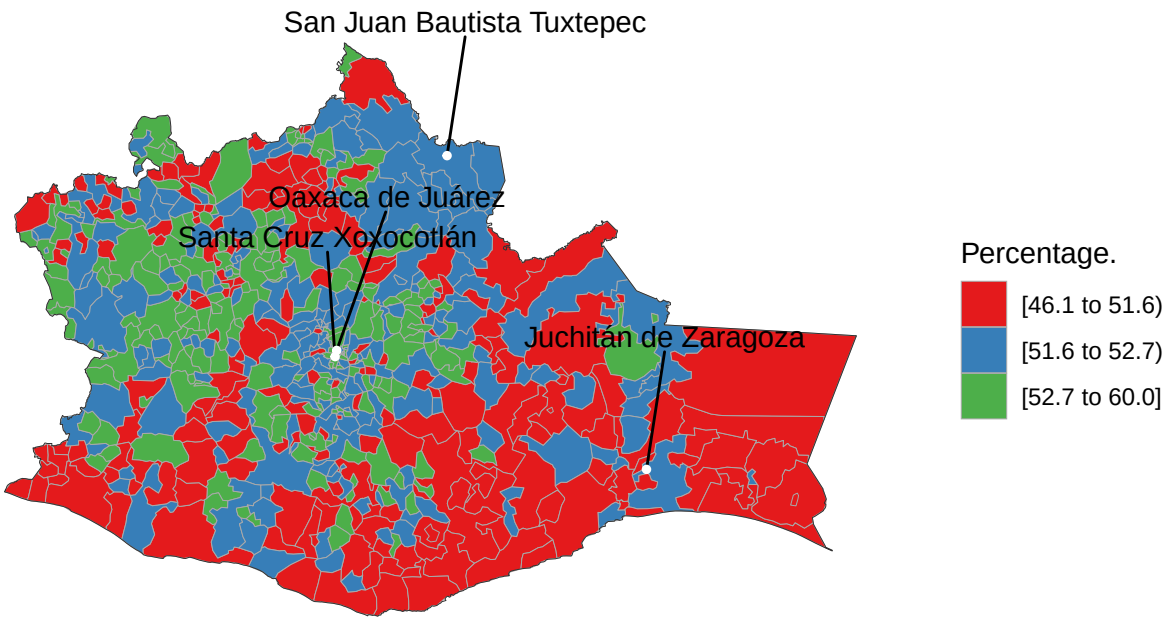
Percentage of female population.



Not sure where are the mail cities?

```
st <- subset(df_mxmunicipio_2020, state_name %in% c("Oaxaca"))
labels <- st
labels$group <- NA
labels <- subset(labels, pop > 1e05)

p + geom_text_repel(data = labels, aes(long, lat, label = municipio_name),
                    nudge_x = 0.1, nudge_y = 0.7) +
  geom_point(data = labels, aes(long, lat), color = "white", size = 1)
```

Percentage of female population.



8 Memes.

See:

- https://www.r-bloggers.com/2020/11/create-bart-simpson-blackboard-memes-with-r/
- http://free-extras.com/images/bart_simpson_chalkboard-5157.htm

9 Covid in México.

9.1 Time series: states.

Please see:

- https://datos.covid-19.conacyt.mx/
- https://rpubs.com/MichelleReyes_216000/U1A6ISW2

This is for August 13th, 2021:

```
datos <- read.csv("Casos_Diarios_Estado_Nacional_Confirmados_20210813.csv")
```

Analyze Nuevo León and Coahuila.

```
nl <- t(datos[datos$nombre == "NUEVO LEON" ,])
nl <- as.vector(nl)
nl <- nl[4:545]
nl <- as.numeric(nl)
ac_nl <- cumsum(nl)

co <- t(datos[datos$nombre == "COAHUILA" ,])
co <- as.vector(co)
co <- co[4:545]
co <- as.numeric(co)
ac_co <- cumsum(co)

Date <- seq(from = as.Date("2020-02-18"), to = as.Date("2021-08-12"),
            by = "day" )

nlco <- data.frame(Date, nl, co)

ac_nlco <- data.frame(Date, ac_nl ,ac_co)
```

```
tail(nlco)
```

```
##           Date   nl  co
## 537 2021-08-07  480 149
## 538 2021-08-08  407 113
## 539 2021-08-09 1573 332
## 540 2021-08-10 1471 326
## 541 2021-08-11 1089 231
## 542 2021-08-12  475  80
```

There is seasonality evidence in the time series.

```
data1 <- nlco
data1$day <- weekdays(nlco$Date)
tail(data1, 20)
```

```
##           Date   nl  co       day
## 523 2021-07-24  385  86    sábado
## 524 2021-07-25  290  51   domingo
## 525 2021-07-26 1138 259     lunes
## 526 2021-07-27 1211 265    martes
## 527 2021-07-28 1099 305 miércoles
## 528 2021-07-29 1037 267    jueves
## 529 2021-07-30 1144 304   viernes
## 530 2021-07-31  424 131    sábado
## 531 2021-08-01  359  81   domingo
## 532 2021-08-02 1517 350     lunes
## 533 2021-08-03 1396 328    martes
## 534 2021-08-04 1294 297 miércoles
## 535 2021-08-05 1383 303    jueves
## 536 2021-08-06 1316 291   viernes
## 537 2021-08-07  480 149    sábado
## 538 2021-08-08  407 113   domingo
## 539 2021-08-09 1573 332     lunes
## 540 2021-08-10 1471 326    martes
## 541 2021-08-11 1089 231 miércoles
## 542 2021-08-12  475  80    jueves
```

Quantify the seasonality.

```
data1_weekend <- data1 |>
  filter(day == "sábado" | day == "domingo") |>
  select(nl, co) |>
  summary()
print("Weekend")
```

```
## [1] "Weekend"
```

```
data1_weekend
```

```
##        nl              co
##  Min.   :  0.0   Min.   :  0.00
##  1st Qu.: 30.0   1st Qu.: 11.00
##  Median :113.0   Median : 47.50
##  Mean   :145.4   Mean   : 70.88
```

```
##  3rd Qu.:231.0   3rd Qu.:124.50
##  Max.   :531.0   Max.   :233.00
```

```
data1_weekday <- data1 |>
  filter(day != "sábado" | day != "domingo") |>
  select(nl, co) |>
  summary()
print("Weekday")
```

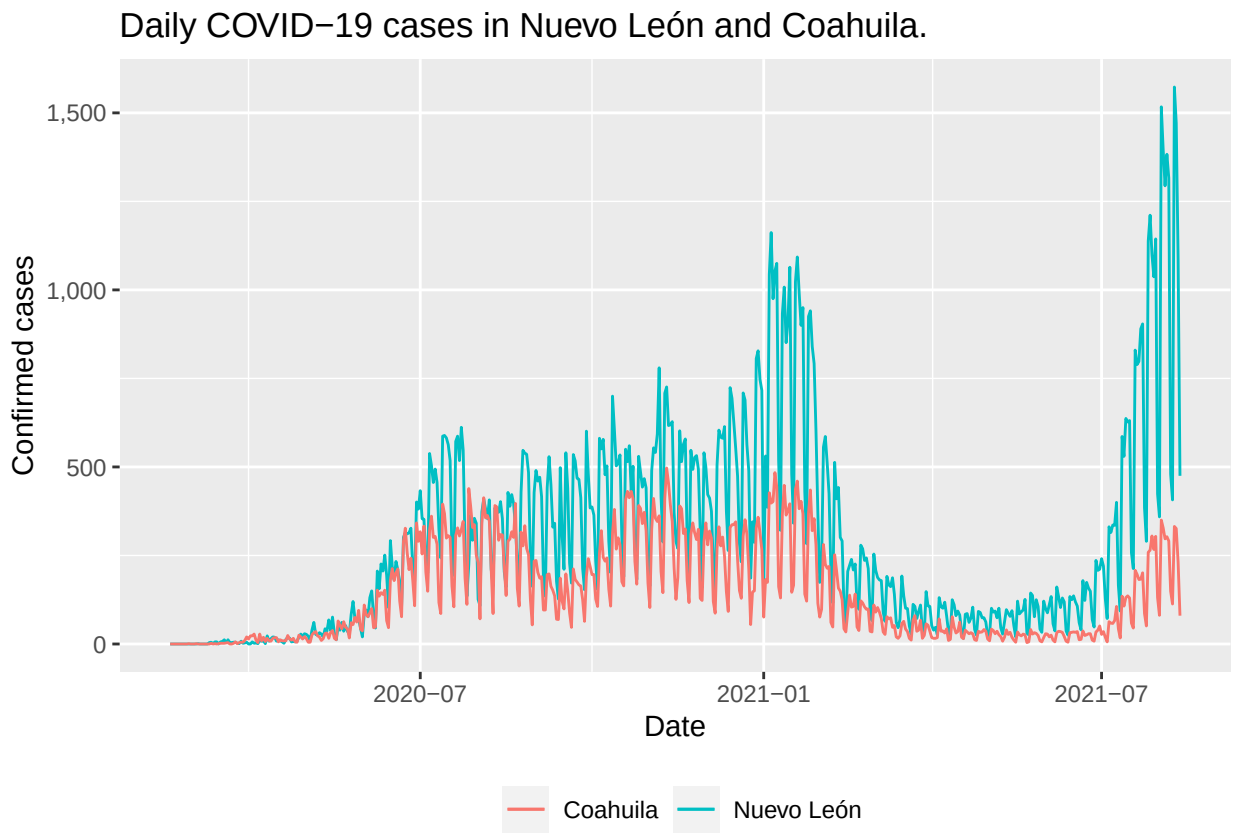
```
## [1] "Weekday"
```

```
data1_weekday
```

```
##        nl               co
##  Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  64.5   1st Qu.: 27.0
##  Median : 201.5   Median : 95.5
##  Mean   : 291.1   Mean   :141.2
##  3rd Qu.: 463.8   3rd Qu.:252.0
##  Max.   :1573.0   Max.   :497.0
```

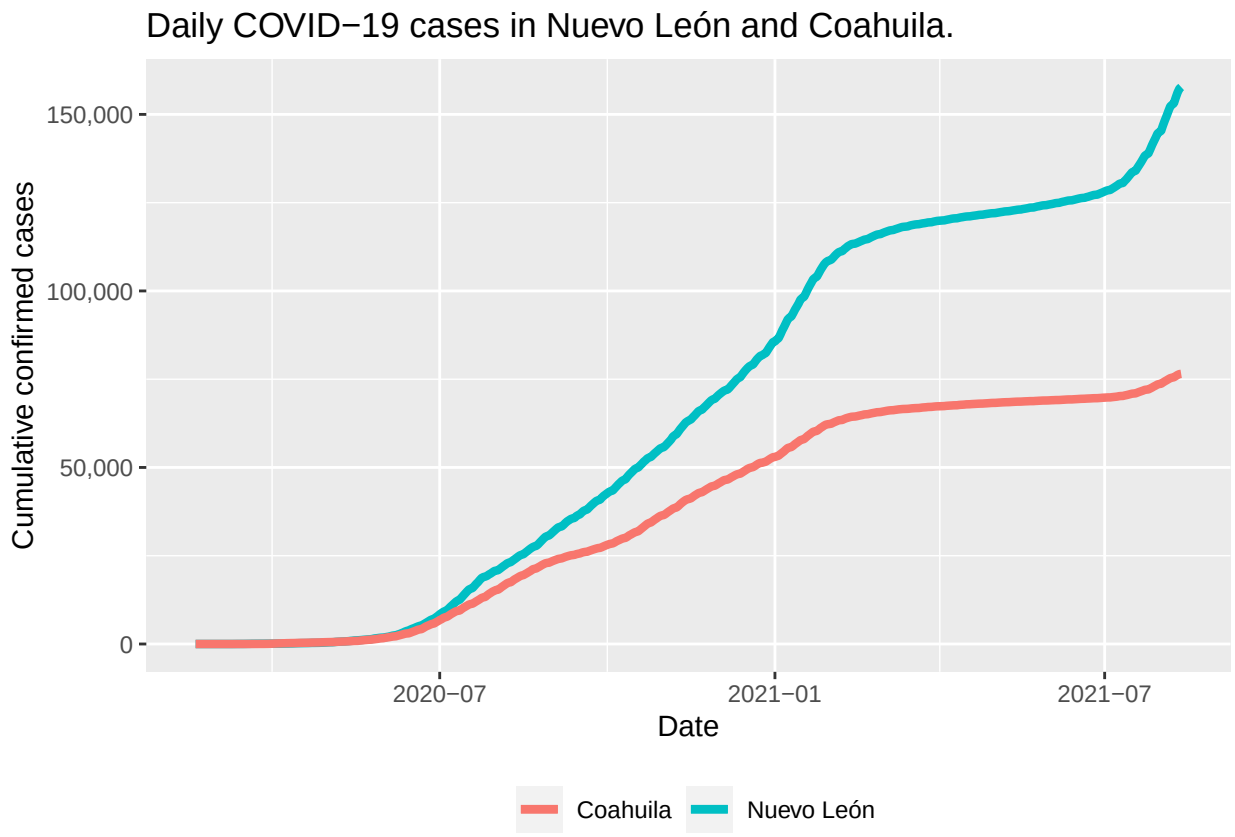
Confirmed cases.

```
ggplot(data = nlco) +
  geom_line(aes(Date, nl, colour="Nuevo León")) +
  geom_line(aes(Date, co, colour="Coahuila")) +
  xlab("Date") +
  ylab("Confirmed cases") +
  scale_y_continuous(labels = comma) +
  ggtitle("Daily COVID-19 cases in Nuevo León and Coahuila.") +
  theme(legend.title = element_blank(), legend.position = "bottom")
```



Cumulative confirmed cases.

```
ggplot(data = nlco) +
  geom_line(aes(Date, ac_nl, colour = "Nuevo León"), size = 1.5) +
  geom_line(aes(Date, ac_co, colour = "Coahuila"), size = 1.5) +
  xlab("Date") +
  ylab("Cumulative confirmed cases") +
  ggtitle("Daily COVID-19 cases in Nuevo León and Coahuila.") +
  scale_y_continuous(labels = comma) +
  theme(legend.title = element_blank(), legend.position = "bottom")
```



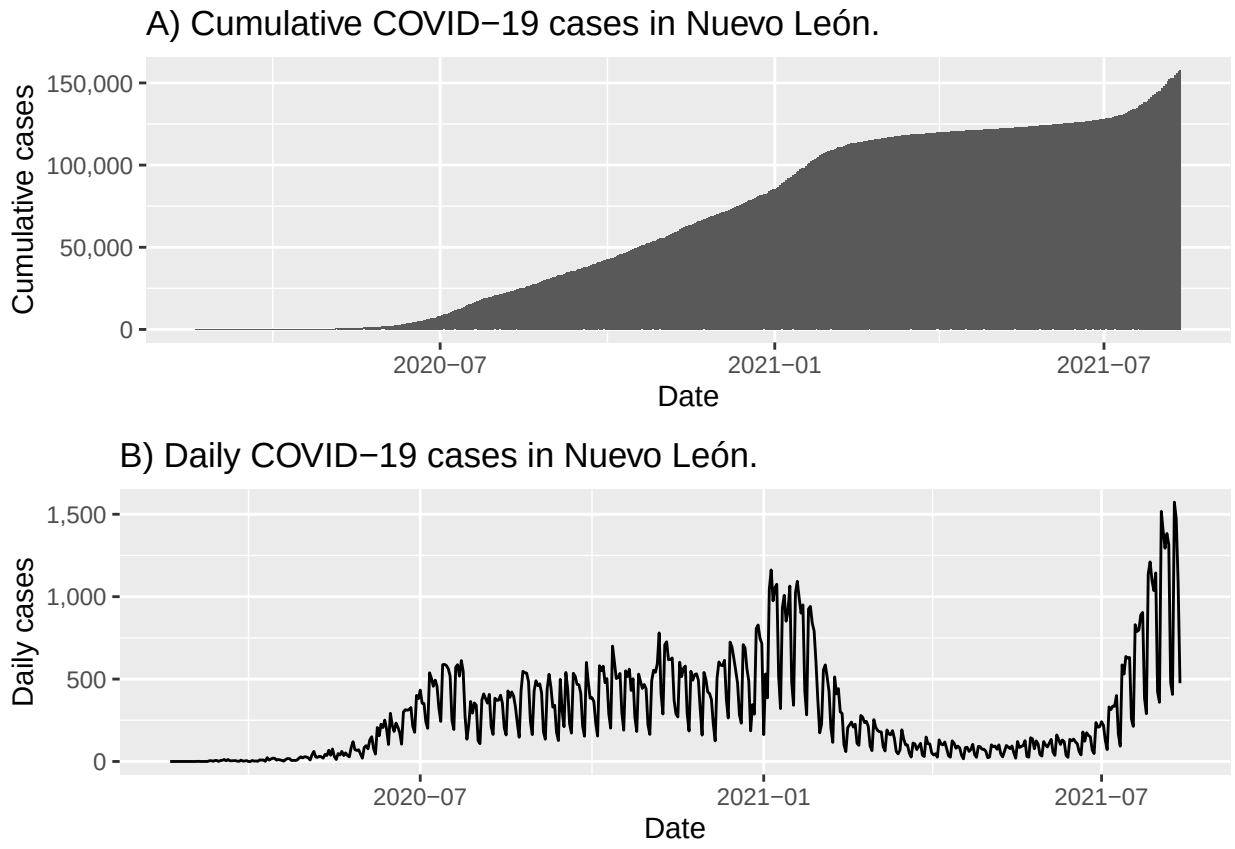

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
nl1 <- data.frame(Date, nl, ac_nl)

g2 <- ggplot(data = nl1) +
  geom_col(aes(Date, ac_nl)) +
  xlab("Date") +
  ylab("Cumulative cases") +
  scale_y_continuous(labels = comma) +
  ggtitle("A) Cumulative COVID-19 cases in Nuevo León.")

g3 <- ggplot(data = nl1) +
```

```
  geom_line(aes(Date, nl)) +
  xlab("Date") +
  ylab("Daily cases") +
  scale_y_continuous(labels = comma) +
  ggtitle("B) Daily COVID-19 cases in Nuevo León.")

grid.arrange(g2, g3)
```



9.2 Cross section: open data.

- https://www.gob.mx/salud/documentos/datos-abiertos-152127
- https://gist.github.com/diegovalle/9e72ecc855f720aa645e33494d6efcb8

This is for August 13th, 2021.

Read more than 9 million observations of 40 variables.

```
df <- read_csv("210813COVID19MEXICO.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   FECHA_ACTUALIZACION = col_date(format = ""),
##   ID_REGISTRO = col_character(),
##   ENTIDAD_UM = col_character(),
##   ENTIDAD_NAC = col_character(),
##   ENTIDAD_RES = col_character(),
##   MUNICIPIO_RES = col_character(),
##   FECHA_INGRESO = col_date(format = ""),
##   FECHA_SINTOMAS = col_date(format = ""),
##   FECHA_DEF = col_date(format = ""),
##   PAIS_NACIONALIDAD = col_character(),
##   PAIS_ORIGEN = col_character()
## )
## i Use `spec()` for the full column specifications.

## Warning: 8703591 parsing failures.
## row       col   expected     actual                       file
##   1 FECHA_DEF valid date 9999-99-99 '210813COVID19MEXICO.csv'
##   3 FECHA_DEF valid date 9999-99-99 '210813COVID19MEXICO.csv'
##   4 FECHA_DEF valid date 9999-99-99 '210813COVID19MEXICO.csv'
##   5 FECHA_DEF valid date 9999-99-99 '210813COVID19MEXICO.csv'
##   6 FECHA_DEF valid date 9999-99-99 '210813COVID19MEXICO.csv'
## ... ......... .......... .......... ........................
## See problems(...) for more details.
```

```
str(df)
```

```
## spec_tbl_df [9,022,864 x 40] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ FECHA_ACTUALIZACION  : Date[1:9022864], format: "2021-08-13" "2021-08-13" ...
##  $ ID_REGISTRO          : chr [1:9022864] "z482b8" "z49a69" "z23d9d" "z24953" ...
##  $ ORIGEN               : num [1:9022864] 2 1 1 1 2 1 1 2 2 1 ...
##  $ SECTOR               : num [1:9022864] 12 12 12 12 12 12 12 12 12 12 ...
##  $ ENTIDAD_UM           : chr [1:9022864] "09" "23" "22" "09" ...
##  $ SEXO                 : num [1:9022864] 2 1 2 1 2 1 1 1 2 1 ...
##  $ ENTIDAD_NAC          : chr [1:9022864] "09" "23" "24" "09" ...
```

```
##  $ ENTIDAD_RES          : chr [1:9022864] "09" "23" "22" "09" ...
##  $ MUNICIPIO_RES        : chr [1:9022864] "012" "004" "009" "010" ...
##  $ TIPO_PACIENTE        : num [1:9022864] 1 2 1 1 1 1 1 1 1 1 ...
##  $ FECHA_INGRESO        : Date[1:9022864], format: "2020-10-16" "2020-07-20" ...
##  $ FECHA_SINTOMAS       : Date[1:9022864], format: "2020-10-16" "2020-07-17" ...
##  $ FECHA_DEF            : Date[1:9022864], format: NA "2020-07-21" ...
##  $ INTUBADO             : num [1:9022864] 97 1 97 97 97 97 97 97 97 97 ...
##  $ NEUMONIA             : num [1:9022864] 2 1 2 2 2 2 2 2 2 2 ...
##  $ EDAD                 : num [1:9022864] 41 66 29 40 34 48 60 20 47 40 ...
##  $ NACIONALIDAD         : num [1:9022864] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EMBARAZO             : num [1:9022864] 97 2 97 98 97 2 2 2 97 98 ...
##  $ HABLA_LENGUA_INDIG   : num [1:9022864] 99 2 2 99 2 2 2 2 2 99 ...
##  $ INDIGENA             : num [1:9022864] 99 2 2 99 2 2 2 2 2 99 ...
##  $ DIABETES             : num [1:9022864] 2 1 2 2 2 1 2 2 1 2 ...
##  $ EPOC                 : num [1:9022864] 2 2 2 2 1 2 2 2 2 2 ...
##  $ ASMA                 : num [1:9022864] 2 2 2 2 1 2 2 2 2 2 ...
##  $ INMUSUPR             : num [1:9022864] 2 2 2 2 2 2 2 2 2 2 ...
##  $ HIPERTENSION         : num [1:9022864] 2 1 2 2 2 2 2 2 2 2 ...
##  $ OTRA_COM             : num [1:9022864] 2 2 2 2 2 2 2 2 2 2 ...
##  $ CARDIOVASCULAR       : num [1:9022864] 2 2 2 2 2 2 2 2 2 1 ...
##  $ OBESIDAD             : num [1:9022864] 2 1 98 2 2 1 1 2 2 2 ...
##  $ RENAL_CRONICA        : num [1:9022864] 2 2 2 2 2 2 2 2 2 2 ...
##  $ TABAQUISMO           : num [1:9022864] 2 2 2 2 2 2 2 2 2 2 ...
##  $ OTRO_CASO            : num [1:9022864] 2 1 2 1 2 99 2 2 2 1 ...
##  $ TOMA_MUESTRA_LAB     : num [1:9022864] 2 2 2 1 2 1 1 2 1 1 ...
##  $ RESULTADO_LAB        : num [1:9022864] 97 97 97 2 97 4 2 97 1 2 ...
##  $ TOMA_MUESTRA_ANTIGENO: num [1:9022864] 2 2 2 2 2 2 2 1 2 2 ...
##  $ RESULTADO_ANTIGENO   : num [1:9022864] 97 97 97 97 97 97 97 2 97 97 ...
##  $ CLASIFICACION_FINAL  : num [1:9022864] 1 2 6 7 6 5 7 7 3 7 ...
##  $ MIGRANTE             : num [1:9022864] 99 99 99 99 99 99 99 99 99 99 ...
##  $ PAIS_NACIONALIDAD    : chr [1:9022864] "México" "México" "México" "México" ...
##  $ PAIS_ORIGEN          : chr [1:9022864] "97" "97" "97" "97" ...
##  $ UCI                  : num [1:9022864] 97 1 97 97 97 97 97 97 97 97 ...
##  - attr(*, "problems")= tibble [8,703,591 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ row     : int [1:8703591] 1 3 4 5 6 7 8 9 10 11 ...
##   ..$ col     : chr [1:8703591] "FECHA_DEF" "FECHA_DEF" "FECHA_DEF" "FECHA_DEF" ...
```

```
##   ..$ expected: chr [1:8703591] "valid date" "valid date" "valid date" "valid date" .
##   ..$ actual  : chr [1:8703591] "9999-99-99" "9999-99-99" "9999-99-99" "9999-99-99" .
##   ..$ file    : chr [1:8703591] "'210813COVID19MEXICO.csv'" "'210813COVID19MEXICO.csv
##  - attr(*, "spec")=
##   .. cols(
##   ..   FECHA_ACTUALIZACION = col_date(format = ""),
##   ..   ID_REGISTRO = col_character(),
##   ..   ORIGEN = col_double(),
##   ..   SECTOR = col_double(),
##   ..   ENTIDAD_UM = col_character(),
##   ..   SEXO = col_double(),
##   ..   ENTIDAD_NAC = col_character(),
##   ..   ENTIDAD_RES = col_character(),
##   ..   MUNICIPIO_RES = col_character(),
##   ..   TIPO_PACIENTE = col_double(),
##   ..   FECHA_INGRESO = col_date(format = ""),
##   ..   FECHA_SINTOMAS = col_date(format = ""),
##   ..   FECHA_DEF = col_date(format = ""),
##   ..   INTUBADO = col_double(),
##   ..   NEUMONIA = col_double(),
##   ..   EDAD = col_double(),
##   ..   NACIONALIDAD = col_double(),
##   ..   EMBARAZO = col_double(),
##   ..   HABLA_LENGUA_INDIG = col_double(),
##   ..   INDIGENA = col_double(),
##   ..   DIABETES = col_double(),
##   ..   EPOC = col_double(),
##   ..   ASMA = col_double(),
##   ..   INMUSUPR = col_double(),
##   ..   HIPERTENSION = col_double(),
##   ..   OTRA_COM = col_double(),
##   ..   CARDIOVASCULAR = col_double(),
##   ..   OBESIDAD = col_double(),
##   ..   RENAL_CRONICA = col_double(),
##   ..   TABAQUISMO = col_double(),
##   ..   OTRO_CASO = col_double(),
```

```
##   ..   TOMA_MUESTRA_LAB = col_double(),
##   ..   RESULTADO_LAB = col_double(),
##   ..   TOMA_MUESTRA_ANTIGENO = col_double(),
##   ..   RESULTADO_ANTIGENO = col_double(),
##   ..   CLASIFICACION_FINAL = col_double(),
##   ..   MIGRANTE = col_double(),
##   ..   PAIS_NACIONALIDAD = col_character(),
##   ..   PAIS_ORIGEN = col_character(),
##   ..   UCI = col_double()
##   .. )
```

We have 360,914,560 observations.

```
9022864*40
```

```
## [1] 360914560
```

First values:

```
head(df)
```

```
## # A tibble: 6 x 40
##   FECHA_ACTUALIZACION ID_REGISTRO ORIGEN SECTOR ENTIDAD_UM  SEXO ENTIDAD_NAC
##   <date>              <chr>        <dbl>  <dbl> <chr>      <dbl> <chr>
## 1 2021-08-13          z482b8           2     12 09             2 09
## 2 2021-08-13          z49a69           1     12 23             1 23
## 3 2021-08-13          z23d9d           1     12 22             2 24
## 4 2021-08-13          z24953           1     12 09             1 09
## 5 2021-08-13          zz8e77           2     12 09             2 09
## 6 2021-08-13          z1b0d1           1     12 01             1 01
## # ... with 33 more variables: ENTIDAD_RES <chr>, MUNICIPIO_RES <chr>,
## #   TIPO_PACIENTE <dbl>, FECHA_INGRESO <date>, FECHA_SINTOMAS <date>,
## #   FECHA_DEF <date>, INTUBADO <dbl>, NEUMONIA <dbl>, EDAD <dbl>,
## #   NACIONALIDAD <dbl>, EMBARAZO <dbl>, HABLA_LENGUA_INDIG <dbl>,
## #   INDIGENA <dbl>, DIABETES <dbl>, EPOC <dbl>, ASMA <dbl>, INMUSUPR <dbl>,
## #   HIPERTENSION <dbl>, OTRA_COM <dbl>, CARDIOVASCULAR <dbl>, OBESIDAD <dbl>,
## #   RENAL_CRONICA <dbl>, TABAQUISMO <dbl>, OTRO_CASO <dbl>,
## #   TOMA_MUESTRA_LAB <dbl>, RESULTADO_LAB <dbl>, TOMA_MUESTRA_ANTIGENO <dbl>,
## #   RESULTADO_ANTIGENO <dbl>, CLASIFICACION_FINAL <dbl>, MIGRANTE <dbl>,
```

```
## #   PAIS_NACIONALIDAD <chr>, PAIS_ORIGEN <chr>, UCI <dbl>
```

We are interested on the confirmed lab cases.

```
pos <- filter(df, RESULTADO_LAB == 1)
head(pos)
```

```
## # A tibble: 6 x 40
##   FECHA_ACTUALIZACION ID_REGISTRO ORIGEN SECTOR ENTIDAD_UM  SEXO ENTIDAD_NAC
##   <date>              <chr>        <dbl>  <dbl> <chr>      <dbl> <chr>
## 1 2021-08-13          z33a15           2     12 12             2 12
## 2 2021-08-13          z4f06b           1      6 22             2 15
## 3 2021-08-13          z2770b           1     12 09             1 09
## 4 2021-08-13          z166d5           1     12 01             1 01
## 5 2021-08-13          zz8e57           1     12 31             1 31
## 6 2021-08-13          z4494e           2     12 08             1 08
## # ... with 33 more variables: ENTIDAD_RES <chr>, MUNICIPIO_RES <chr>,
## #   TIPO_PACIENTE <dbl>, FECHA_INGRESO <date>, FECHA_SINTOMAS <date>,
## #   FECHA_DEF <date>, INTUBADO <dbl>, NEUMONIA <dbl>, EDAD <dbl>,
## #   NACIONALIDAD <dbl>, EMBARAZO <dbl>, HABLA_LENGUA_INDIG <dbl>,
## #   INDIGENA <dbl>, DIABETES <dbl>, EPOC <dbl>, ASMA <dbl>, INMUSUPR <dbl>,
## #   HIPERTENSION <dbl>, OTRA_COM <dbl>, CARDIOVASCULAR <dbl>, OBESIDAD <dbl>,
## #   RENAL_CRONICA <dbl>, TABAQUISMO <dbl>, OTRO_CASO <dbl>,
## #   TOMA_MUESTRA_LAB <dbl>, RESULTADO_LAB <dbl>, TOMA_MUESTRA_ANTIGENO <dbl>,
## #   RESULTADO_ANTIGENO <dbl>, CLASIFICACION_FINAL <dbl>, MIGRANTE <dbl>,
## #   PAIS_NACIONALIDAD <chr>, PAIS_ORIGEN <chr>, UCI <dbl>
```

I know Nuevo León is 19 and Monterrey is 039.

```
mty <- pos |>
  filter(MUNICIPIO_RES == "039" & ENTIDAD_RES == "19")
mty
```

```
## # A tibble: 35,290 x 40
##    FECHA_ACTUALIZACION ID_REGISTRO ORIGEN SECTOR ENTIDAD_UM  SEXO ENTIDAD_NAC
##    <date>              <chr>        <dbl>  <dbl> <chr>      <dbl> <chr>
##  1 2021-08-13          z22bc2           2     13 19             2 19
##  2 2021-08-13          0a8c40           1     12 19             2 19
##  3 2021-08-13          07feb9           2     12 19             2 19
```

```
##  4 2021-08-13          1545cb          2      9 19             2 24
##  5 2021-08-13          1e7a91          2      4 19             2 19
##  6 2021-08-13          19a7a6          1      4 19             2 19
##  7 2021-08-13          1863b0          2      9 19             2 19
##  8 2021-08-13          1521b7          1     13 19             2 09
##  9 2021-08-13          057bd8          1     13 19             2 19
## 10 2021-08-13          19adbb          1     13 19             1 19
## # ... with 35,280 more rows, and 33 more variables: ENTIDAD_RES <chr>,
## #   MUNICIPIO_RES <chr>, TIPO_PACIENTE <dbl>, FECHA_INGRESO <date>,
## #   FECHA_SINTOMAS <date>, FECHA_DEF <date>, INTUBADO <dbl>, NEUMONIA <dbl>,
## #   EDAD <dbl>, NACIONALIDAD <dbl>, EMBARAZO <dbl>, HABLA_LENGUA_INDIG <dbl>,
## #   INDIGENA <dbl>, DIABETES <dbl>, EPOC <dbl>, ASMA <dbl>, INMUSUPR <dbl>,
## #   HIPERTENSION <dbl>, OTRA_COM <dbl>, CARDIOVASCULAR <dbl>, OBESIDAD <dbl>,
## #   RENAL_CRONICA <dbl>, TABAQUISMO <dbl>, OTRO_CASO <dbl>,
## #   TOMA_MUESTRA_LAB <dbl>, RESULTADO_LAB <dbl>, TOMA_MUESTRA_ANTIGENO <dbl>,
## #   RESULTADO_ANTIGENO <dbl>, CLASIFICACION_FINAL <dbl>, MIGRANTE <dbl>,
## #   PAIS_NACIONALIDAD <chr>, PAIS_ORIGEN <chr>, UCI <dbl>
```

Here we incorporate Diego Valle's package.

```
# There are some missing values.
pos1 <- pos |>
  filter(MUNICIPIO_RES != "999")

pos1$region <- str_mxmunicipio(pos1$ENTIDAD_RES, pos1$MUNICIPIO_RES)
```

```
## Warning in str_mxmunicipio(pos1$ENTIDAD_RES, pos1$MUNICIPIO_RES): Invalid codes
## detected
```

We have 2,468 municipalities in Mexico.

```
muns <- pos1 |>
  group_by(region) |>
  tally()
```

Join open data with Diego Valle database.

```
muns <- left_join(muns, df_mxmunicipio, by = "region")
muns$value <-  muns$n
#muns$value <-  round(muns$n / muns$pop * 10^5)
```

```
#muns$value <- if_else(muns$value2 > 400, 400, muns$value2)
muns$name <- paste(muns$state_name, muns$municipio_name)
```

```
library(stringr)
library(ggrepel)
cities <- filter(muns, name %in% c("Coahuila Monclova",
                                   "Baja California Sur Los Cabos",
                                   "Ciudad de México Cuajimalpa de Morelos",
                                   "Quintana Roo Benito Juárez",
                                   "Sinaloa Culiacán",
                                   "Tabasco Centro",
                                   "Baja California Mexicali",
                                   "Yucatán Mérida",
                                   "Puebla Puebla",
                                   "Sonora Sáric",
                                   "Guerrero Acapulco de Juárez",
                                   "Chihuahua Juárez",
                                   "Coahuila Piedras Negras",
                                   "Michoacán Lázaro Cárdenas",
                                   "Oaxaca Santa María Huatulco",
                                   "Colima Manzanillo",
                                   "Jalisco Puerto Vallarta"))
cities$municipio_name <- str_replace(cities$municipio_name,
                                     "Cuajimalpa de Morelos",
                                     "Cuajimalpa")
cities$municipio_name <- str_replace(cities$municipio_name,
                                     "Santa María Huatulco",
                                     "Huatulco")
cities$municipio_name <- str_replace(cities$municipio_name,
                                     "Benito Juárez",
                                     "Cancún")
cities$municipio_name <- str_replace(cities$municipio_name,
                                     "Juárez",
                                     "Ciudad Juárez")
cities$municipio_name <- str_replace(cities$municipio_name,
                                     "Acapulco de Juárez",
```

```
                                      "Acapulco")
cities$municipio_name <- str_replace(cities$municipio_name,
                                      "Centro",
                                      "Villahermosa")
cities$group <- NA
```

```
p <- mxmunicipio_choropleth(muns,
                        num_colors = 4,
    title = "Casos COVID-19 confirmados (13 de agosto del 2021).",
                        legend = "Valores brutos.") +
  geom_label_repel(data = cities, aes(long, lat, label = municipio_name),
                   size = 3,
                   force = .1, alpha = .8,
                   box.padding = 3.3, label.padding = 0.18) +
  theme(legend.key.size = unit(1, "cm")) +
  theme(plot.title = element_text(size = 10)) +
  theme(legend.title = element_blank(), legend.position = "bottom")
```

```
## Warning in str_mxmunicipio(df$region): Invalid codes detected
```

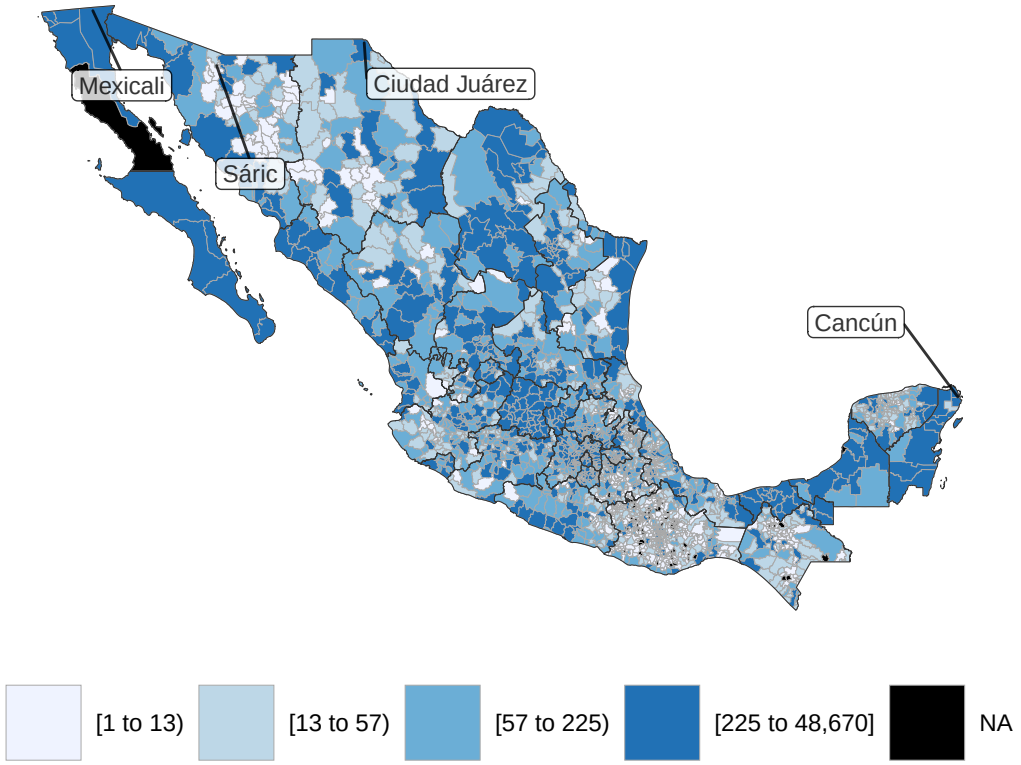
```
## Warning in super$initialize(mxmunicipio.map, user.df): Your data.frame contains
## the following regions which are not mappable: 09106, 17056, 17063
```

```
## Warning in self$bind(): The following regions were missing and are being set to
## NA: 02006, 04012, 07039, 07090, 07115, 07118, 07120, 07121, 07125, 17036, 20008,
## 20029, 20074, 20098, 20100, 20101, 20106, 20110, 20113, 20114, 20119, 20120,
## 20122, 20128, 20142, 20151, 20155, 20170, 20172, 20173, 20181, 20186, 20187,
## 20191, 20196, 20199, 20209, 20213, 20218, 20220, 20222, 20223, 20225, 20228,
## 20253, 20257, 20264, 20267, 20274, 20280, 20282, 20311, 20313, 20316, 20317,
## 20321, 20322, 20329, 20331, 20332, 20335, 20336, 20354, 20355, 20357, 20361,
## 20373, 20376, 20382, 20383, 20395, 20408, 20423, 20430, 20433, 20436, 20442,
## 20444, 20448, 20463, 20464, 20471, 20473, 20476, 20479, 20480, 20490, 20491,
## 20495, 20503, 20504, 20510, 20512, 20514, 20521, 20522, 20532, 20541, 20556,
## 20564, 21024, 21036, 21052, 30195
```

```
p
```

```
## Warning: ggrepel: 13 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

Casos COVID−19 confirmados (13 de agosto del 2021).



A clearer picture.

```
p <- mxmunicipio_choropleth(muns,
    title = "Casos COVID-19 confirmados (13 de agosto del 2021).",
   legend = "Valores brutos.", num_colors = 4)
```

```
## Warning in str_mxmunicipio(df$region): Invalid codes detected
```

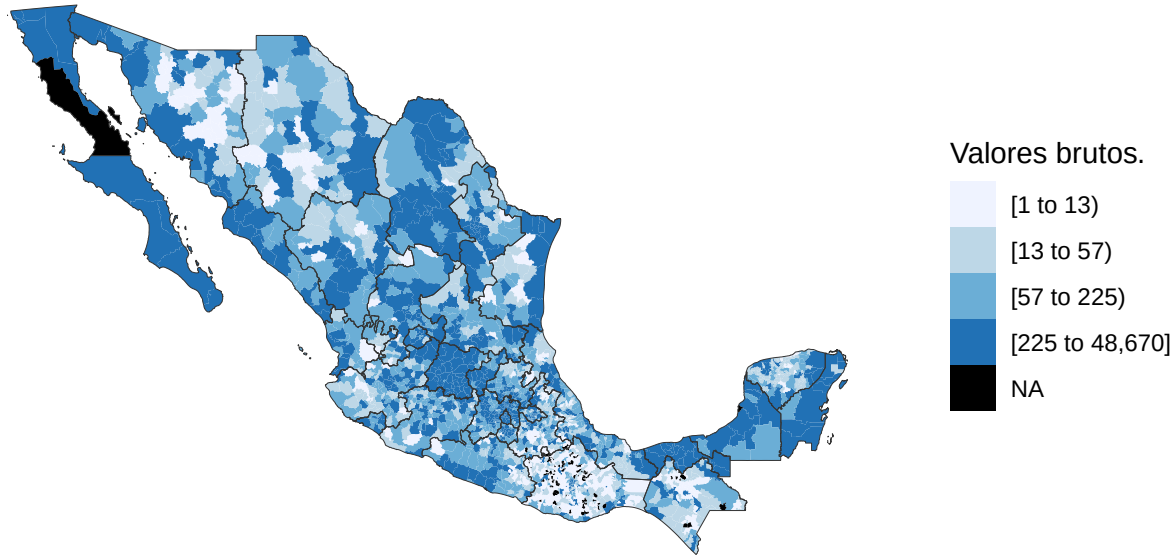
```
## Warning in super$initialize(mxmunicipio.map, user.df): Your data.frame contains
## the following regions which are not mappable: 09106, 17056, 17063
```

```
## Warning in self$bind(): The following regions were missing and are being set to
## NA: 02006, 04012, 07039, 07090, 07115, 07118, 07120, 07121, 07125, 17036, 20008,
## 20029, 20074, 20098, 20100, 20101, 20106, 20110, 20113, 20114, 20119, 20120,
## 20122, 20128, 20142, 20151, 20155, 20170, 20172, 20173, 20181, 20186, 20187,
## 20191, 20196, 20199, 20209, 20213, 20218, 20220, 20222, 20223, 20225, 20228,
## 20253, 20257, 20264, 20267, 20274, 20280, 20282, 20311, 20313, 20316, 20317,
## 20321, 20322, 20329, 20331, 20332, 20335, 20336, 20354, 20355, 20357, 20361,
## 20373, 20376, 20382, 20383, 20395, 20408, 20423, 20430, 20433, 20436, 20442,
```

```
## 20444, 20448, 20463, 20464, 20471, 20473, 20476, 20479, 20480, 20490, 20491,
## 20495, 20503, 20504, 20510, 20512, 20514, 20521, 20522, 20532, 20541, 20556,
## 20564, 21024, 21036, 21052, 30195
```

```
p[["layers"]][[1]][["aes_params"]][["colour"]] <- "transparent"
p
```

Casos COVID−19 confirmados (13 de agosto del 2021).



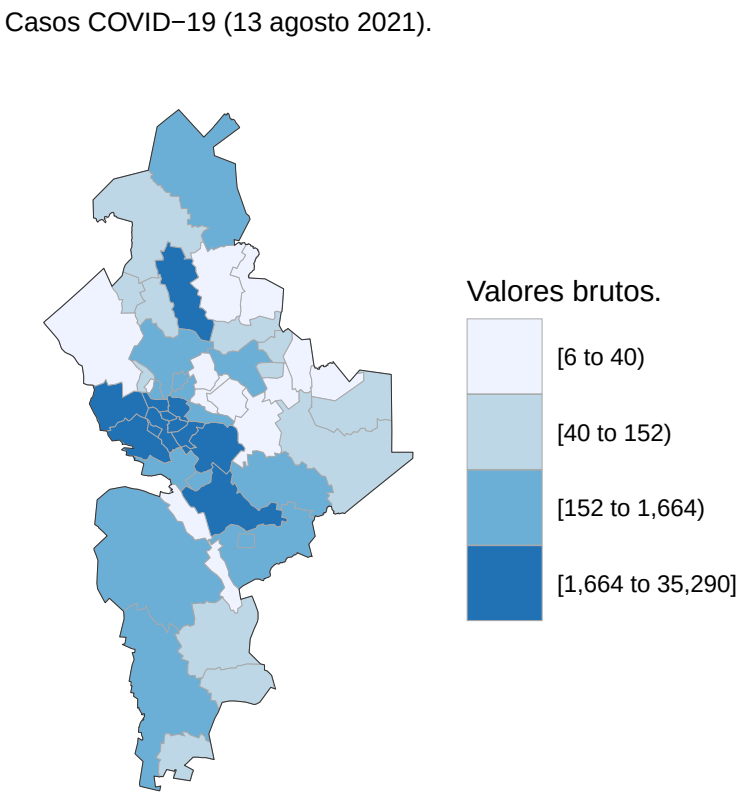
Nuevo León.

```
p <- mxmunicipio_choropleth(muns,
                        num_colors = 4,
                        title = "Casos COVID-19 (13 agosto 2021).",
                        legend = "Valores brutos.",
                        zoom = subset(df_mxmunicipio,
                                      state_name %in% c("Nuevo León"))$region) +
  theme(legend.key.size = unit(1, "cm")) +
  theme(plot.title = element_text(size=10))
```

```
## Warning in str_mxmunicipio(df$region): Invalid codes detected
```

```
## Warning in super$initialize(mxmunicipio.map, user.df): Your data.frame contains
## the following regions which are not mappable: 09106, 17056, 17063
```

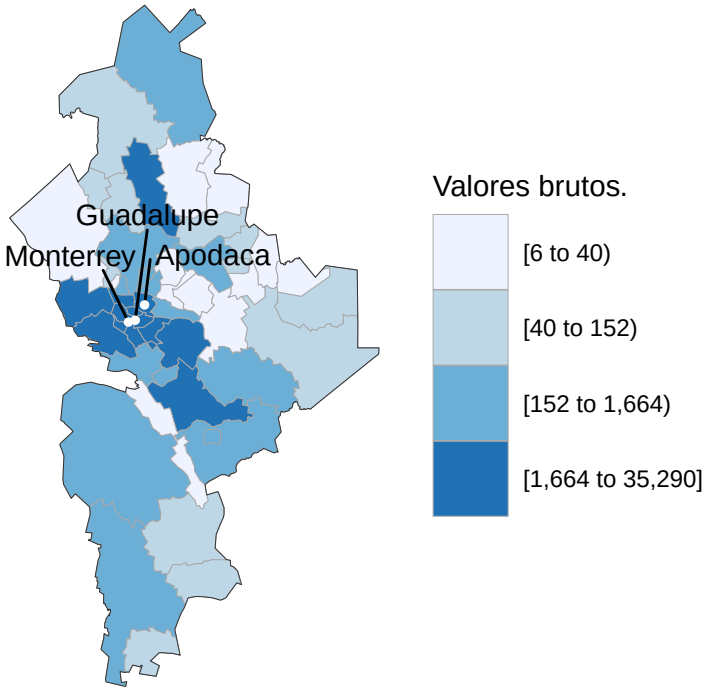
```
p
```



With biggest cities.

```
st <- subset(df_mxmunicipio_2020, state_name %in% c("Nuevo León"))
labels <- st
labels$group <- NA
labels <- subset(labels, pop > 5e05)

p + geom_text_repel(data = labels, aes(long, lat, label = municipio_name),
                    nudge_x = 0.1, nudge_y = 0.7) +
  geom_point(data = labels, aes(long, lat), color = "white", size = 1)
```

Casos COVID−19 (13 agosto 2021).



Nice.

```
a <- toc()
```

```
## 236.11 sec elapsed
```

This document took 236.11 seconds to compile in Rmarkdown.

9.3 Use Rmarkdown.

This document was made with Rmarkdown.

Illustration by Allison Horst.

References.

Jennifer Bryan. *Happy Git and GitHub for the UseR*. GitHub, 2018. URL https://happygitwithr.com/.

Andrie De Vries and Joris Meys. *R for Dummies*. John Wiley & Sons, 2nd edition, 2015. URL http://sgpwe.izt.uam.mx/files/users/uami/gma/R_for_dummies.pdf.

Garrett Grolemund and Hadley Wickham. *R for Data Science*. 2018. URL https://r4ds.had.co.nz/.

Christoph Hanck, Martin Arnold, Alexander Gerber, and Martin Schmelzer. *Introduction to Econometrics with R*. 2020. URL https://www.econometrics-with-r.org/.

F. Heiss. *Using R for Introductory Econometrics*. Independently Published, 2020. ISBN 9798648424364. URL http://www.urfie.net/read/index.html.

Donald Ervin Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

Keith McNulty. *Handbook of Regression Modeling in People Analytics: With Examples in R and Python*. Chapman and Hall/CRC, 2021. URL http://peopleanalytics-regression-book.org/index.html.

Florian Oswald, Vincent Viers, Pierre Villedieu, and Gustave Kennedi. *Introduction to Econometrics with R*. SciencesPo Department of Economics, Paris, France, 2020. URL https://scpoecon.github.io/ScPoEconometrics/.

Roger D Peng. *R Programming for Data Science*. Leanpub, 2016. URL https://bookdown.org/rdpeng/rprogdatascience/.

Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Bookdown, 2016. URL https://ggplot2-book.org/index.html.

Yihui Xie. *Bookdown: Authoring Books and Technical Documents with R Markdown*. CRC Press, 2021. URL https://bookdown.org/yihui/bookdown/.

Yihui Xie, Christophe Dervieux, and Emily Riederer. *R Markdown Cookbook*. CRC Press, 2020. URL https://bookdown.org/yihui/rmarkdown-cookbook/.

Yihui Xie, Joseph J Allaire, and Garrett Grolemund. *R markdown: The Definitive Guide*. CRC Press, 2021. URL https://bookdown.org/yihui/rmarkdown/.