The rest of this post is more focused on the code.  
I will explain how the steps involved in making the above animation, and, in the final section, link to more comprehensive teaching material that will teach not only how to make a multitude of other types of maps in R, but also how to ensure the results are reproducible.

The first stage is to prepare the data.

**Prepare**

Let’s start with the packages.  
If you are new to R, you may want to take a read of [this](https://geocompr.robinlovelace.net/spatial-class.html) first, which points to various resources for setting-up R for geographic data.  
When you have a recent R version and the appropriate packages installed (e.g. by executing devtools::install\_github("geocompr/geocompkg")) the packages can be attached as follows:

library(sf) # spatial data classes

library(rnaturalearth) # world map data

library(readxl) # reading excel files

library(dplyr) # data manipulation

library(tidyr) # data manipulation

library(purrr) # data manipulation

library(cartogram) # cartograms creation

library(tmap) # maps creation

**Data**

To create cartograms of the world population for each year we will need two datasets – one containing spatial data of the world’s countries and one non-spatial with information about the annual population in the world’s countries.  
The first one can be easily downloaded from the [Natural Earth](http://www.naturalearthdata.com/) website, for example using the **rnaturalearth** package:

world\_map = ne\_countries(returnclass = "sf")

The second one is available from the [Gapminder](https://www.gapminder.org/) foundation.  
Gapminder provides [a dataset with population data](https://www.gapminder.org/data/documentation/gd003/) for all countries and world regions from 1800 to 2100.  
We can download and read the dataset using the code below:

if(!dir.exists("data")) dir.create("data")

download.file("http://gapm.io/dl\_pop", destfile = "data/pop1800\_2100.xlsx")

world\_pop = read\_xlsx("data/pop1800\_2100.xlsx", sheet = 7)

**Cleaning**

As always when working with multiple datasets – some data cleaning will be necessary.  
Our world\_map dataset has many columns unnecessary for cartograms creation and we do not need spatial data of Antarctica.  
Let’s get rid of them.  
we can also transform our data into a more appropriate projection[1](https://nowosad.github.io/post/world-pop-change/#fn1).

world\_map = world\_map %>%

select(sovereignt) %>%

filter(sovereignt != "Antarctica") %>%

st\_transform(world\_map, crs = "+proj=robin")

We need to have a common identifier to combine our spatial and non-spatial datasets, for example, names of the countries.  
However, there are inconsistencies between some of the names.  
We need to fix it manually:

world\_pop = world\_pop %>%

mutate(sovereignt = name) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Tanzania", "United Republic of Tanzania")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "United States", "United States of America")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Congo, Dem. Rep.", "Democratic Republic of the Congo")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Bahamas", "The Bahamas")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Serbia", "Republic of Serbia")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Macedonia, FYR", "Macedonia")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Slovak Republic", "Slovakia")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Czech Republic", "Czechia")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Congo, Rep.", "Republic of the Congo")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Kyrgyz Republic", "Kyrgyzstan")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Lao", "Laos")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Swaziland", "eSwatini")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Cote d'Ivoire", "Ivory Coast")) %>%

mutate(sovereignt = replace(sovereignt, sovereignt == "Timor-Leste", "East Timor"))

**Preparing**

Now we can join our two datasets, remove missing values and unimportant columns.  
We also should transform the data into a long format[2](https://nowosad.github.io/post/world-pop-change/#fn2) accepted by the **tmap** package.

world\_data = left\_join(world\_map, world\_pop, by = "sovereignt") %>%

na.omit() %>%

select(-geo, -name, -indicator) %>%

gather(key = "year", value = "population", `1800.0`:`2100.0`) %>%

mutate(year = as.integer(year))

Additionally, we can calculate total global populations in each year:

world\_data = world\_data %>%

group\_by(year) %>%

mutate(total\_pop = sum(as.numeric(population), na.rm = TRUE)) %>%

mutate(title = paste0("Year: ", year, "\nTotal population (mln): ", round(total\_pop/1e6, 2)))

**Subset**

Now, our data contains information about the world population for each year between 1800 and 2100.  
It is possible to use it to create cartograms, however, to reduce calculation time and simplification of the results we will only use data for every 25 years:

world\_data = world\_data %>%

filter(year %in% seq(1800, 2100, by = 25))

**Transforming**

Finally, we are able to create our cartograms.  
We need to split our data into independent annual datasets, create cartograms based on the population variable, and combine all of the cartograms back into one object.

Before we create cartograms of the world, let’s create a single cartogram:

world\_data\_2100 = world\_data %>%

filter(year == 2100)

world\_carto1 = cartogram\_cont(world\_data\_2100, "population", maxSizeError = 1.5)

plot(world\_carto1["population"])

A single cartogram of year 2100.

The resulting plot was with the awesome **sf** package, showing what global populations could look like based on recent trends.  
To make this animated, we need to a) do some more processing (to calculate the cartogram shapes for every year in our sequence) and b) switch to using the awesome **tmap** package for animated map creation.  
Note: this next step may take a few minutes.

# warning: this may make your computer's fan spin!

world\_data\_carto = world\_data %>%

split(.$year) %>%

map(cartogram\_cont, "population", maxSizeError = 1.5) %>%

do.call(rbind, .)

**Visualizing**

Animated maps can be created in a single command using the **tmap** package.  
Below we pass it the data produced in the previous command:

carto\_anim = tm\_shape(world\_data\_carto) +

tm\_polygons("population", title = "Population: ") +

tm\_facets(along = "title", free.coords = FALSE, drop.units = TRUE)

The last step is to save the output object as a .gif file:

tmap\_animation(carto\_anim, filename = "world\_pop\_1800\_2100.gif", delay = 75,

width = 1326, height = 942)

If you enjoyed this tutorial you’re in luck: there’s lots more to learn!  
In addition to chapters on a range of fundamental geographic subjects, there’s an entire chapter on visualization in the online version of **Geocomputation with R**,

**Epilogue**

There are several things that should be kept in mind about this animation.  
Firstly, the animation (as well the Gapminder data) shows the world as if all countries always had the borders they have today.  
This is, of course, untrue, but this simplification allows for the straightforward creation of the animation as well as for an easier understanding of the results.  
Secondly, the most time-consuming part of the animation creation, constructing continuous area cartograms, can be accelerated, with the use of simplified geometries of the world countries (for example using the **rmapshaper** package) or with parallel processing (for example using the **future** or **furrr** packages).  
Thirdly, different time steps can be used – an animation could be created for each year, every five years, etc.  
And finally, cartograms can be animated using a different R package, for example,**gganimate**, or even combine into an interactive app using **shiny**.