In this short blog post I show you how you can use the {gganimate} package to create animations  
from {ggplot2} graphs with data from UNU-WIDER.

**WIID data**

The data is available in Excel and STATA formats, and I thought it was a great opportunity to  
release it as an R package. You can install it with:

Library("wiid4")

Here a short description of the data, taken from UNU-WIDER’s website:

*“The World Income Inequality Database (WIID) presents information on income inequality for  
developed, developing, and transition countries. It provides the most comprehensive set of income  
inequality statistics available and can be downloaded for free.*

*WIID4, released in December 2018, covers 189 countries (including historical entities), with over  
11,000 data points in total. With the current version, the latest observations now reach the year 2017.”*

It was also a good opportunity to play around with the {gganimate} package. This package  
makes it possible to create animations and is an extension to {ggplot2}.

**Preparing the data**

To create a smooth animation, I need to have a cylindrical panel data set; meaning that for each  
country in the data set, there are no missing years. I also chose to focus on certain variables  
only; net income, all the population of the country (instead of just focusing on the economically  
active for instance) as well as all the country itself (and not just the rural areas).

Let’s first load the packages, data and perform the necessary transformations:

library(wiid4)

library(tidyverse)

library(ggrepel)

library(gganimate)

library(brotools)

small\_wiid4 <- wiid4 %>%

mutate(eu = as.character(eu)) %>%

mutate(eu = case\_when(eu == "1" ~ "EU member state",

eu == "0" ~ "Non-EU member state")) %>%

filter(resource == 1, popcovr == 1, areacovr == 1, scale == 2) %>%

group\_by(country) %>%

group\_by(country, year) %>%

filter(quality\_score == max(quality\_score)) %>%

filter(source == min(source)) %>%

filter(!is.na(bottom5)) %>%

group\_by(country) %>%

mutate(flag = ifelse(all(seq(2004, 2016) %in% year), 1, 0)) %>%

filter(flag == 1, year > 2003) %>%

mutate(year = lubridate::ymd(paste0(year, "-01-01")))

For some country and some years, there are several sources of data with varying quality. I only  
keep the highest quality sources with:

group\_by(country, year) %>%

filter(quality\_score == max(quality\_score)) %>%

If there are different sources of equal quality, I give priority to the sources that are the most  
comparable across country (Luxembourg Income Study, LIS data) to less comparable sources with  
(at least that’s my understanding of the source variable):

filter(source == min(source)) %>%

I then remove missing data with:

filter(!is.na(bottom5)) %>%

bottom5 and top5 give the share of income that is controlled by the bottom 5% and top 5%  
respectively. These are the variables that I want to plot.

Finally I keep the years 2004 to 2016, without any interruption with the following line:

mutate(flag = ifelse(all(seq(2004, 2016) %in% year), 1, 0)) %>%

filter(flag == 1, year > 2003) %>%

ifelse(all(seq(2004, 2016) %in% year), 1, 0)) creates a flag that equals 1 only if the years  
2004 to 2016 are present in the data without any interruption. Then I only keep the data from 2004  
on and only where the flag variable equals 1.

In the end, I ended up only with European countries. It would have been interesting to have countries  
from other continents, but apparently only European countries provide data in an annual basis.

**Creating the animation**

To create the animation I first started by creating a static ggplot showing what I wanted;  
a scatter plot of the income by bottom and top 5%. The size of the bubbles should be proportional  
to the GDP of the country (another variable provided in the data). Once the plot looked how I wanted  
I added the lines that are specific to {gganimate}:

labs(title = 'Year: {frame\_time}', x = 'Top 5', y = 'Bottom 5') +

transition\_time(year) +

ease\_aes('linear')

I took this from {gganimate}’s README.

animation <- ggplot(small\_wiid4) +

geom\_point(aes(y = bottom5, x = top5, colour = eu, size = log(gdp\_ppp\_pc\_usd2011))) +

xlim(c(10, 20)) +

geom\_label\_repel(aes(y = bottom5, x = top5, label = country), hjust = 1, nudge\_x = 20) +

theme(legend.position = "bottom") +

theme\_blog() +

scale\_color\_blog() +

labs(title = 'Year: {frame\_time}', x = 'Top 5', y = 'Bottom 5') +

transition\_time(year) +

ease\_aes('linear')

I use geom\_label\_repel to place the countries’ labels on the right of the plot. If I don’t do  
this, the labels of the countries would be floating around and the animation would be unreadable.

I then spent some time trying to render a nice webm instead of a gif. It took some trial and error  
and I am still not entirely satisfied with the result, but here is the code to render the animation:

animate(animation, renderer = ffmpeg\_renderer(options = list(s = "864x480",

vcodec = "libvpx-vp9",

crf = "15",

b = "1600k",

vf = "setpts=5\*PTS")))

The option vf = "setpts=5\*PTS" is important because it slows the video down, so we can actually  
see something. crf = "15" is the quality of the video (lower is better), b = "1600k" is the  
bitrate, and vcodec = "libvpx-vp9" is the codec I use. The video you saw at the top of this  
post is the result.

I would have preferred if the video was smoother, which should be possible by creating more frames.  
I did not find such an option in {gganimate}, and perhaps there is none, at least for now.

In any case {gganimate} is pretty nice to play with, and I’ll definitely use it more!

**Update**

Silly me! It turns out thate the animate() function has arguments that can control the number of frames  
and the duration, without needing to pass options to the renderer. I was looking at options for the  
renderer only, without having read the documentation of the animate() function. It turns out that  
you can pass several arguments to the animate() function; for example, here is how you  
can make a GIF that lasts for 20 seconds running and 20 frames per second, pausing for 5  
frames at the end and then restarting:

animate(animation, nframes = 400, duration = 20, fps = 20, end\_pause = 5, rewind = TRUE)

I guess that you should only pass options to the renderer if you really need fine-grained control.

This took around 2 minutes to finish. You can use the same options with the ffmpeg renderer too.  
Here is what the gif looks like:

Much, much smoother!