Visualisation: Reproducible and re-usable plots

By Radovan Bast, UiT The Arctic University of Norway
ORCID: 0000-0002-7658-1847
Date: February 2022


In this short video I will talk about reproducible and reusable plots, but I would like to start by connecting reproducibility with the FAIR principles.

The FAIR principles have been mentioned in a different video in the series. When we produce and share data and software, and make data visualisation, plots and figures, we try to make these FAIR. We try to make these Findable, Accessible, Interoperable and Reusable. On this slide I also write a few typical problems that we can anticipate when working with data visualisation. Here are typical questions which you may have already asked, or you might have gotten these questions: On which of my external hard drives is my data visualisation? Can you please give me access to the plotting scripts? In terms of interoperability: How can I convert this file format to the other file format? And finally in terms of reusability, I wish I could reuse this visualisation pipeline, which I developed some time ago and now I would like to use it for my new data!

So FAIR principles are really good for others, but they are also really good for your future self. So please ask yourself when creating plots: What problems do you anticipate when recreating this plot 12 months or later in the future? And one recommendation that I can give to everybody, please document all the tools and all the dependencies that you have used to generate plots together with their versions.

And now I would like to mention a couple of tools that help with this, that make this easier, and a couple of tools and languages which are very popular to get plots which are reproducible and reusable. One of these tools is Jupyter notebooks. They are very popular in the Python community. This is a web tool where you can collect Python codes that generate plots, texts, equations, figures etc. All in the same place and creating a computational narrative where we can execute code. We can see the result in one place. Then we can modify and repeat it in a kind of iterative conversation between researcher and data. Then we can share these notebooks.
The name Jupyter derives from Julia, Python and R. These are programming languages. Today Jupyter exists for dozens of programming languages. So this is not limited to Python.

What many research groups do with Jupyter notebooks, is that they can actually share the whole workflow. Here is a very nice example of a research group that was working on the discovery of gravitational waves. They made their data visualisation pipeline and their data publicly available on GitHub. Here is the notebook. They have documented their dependencies. There exist tools like Binder, which make it possible for me to rerun their entire workflow directly in the browser. I will start this, as an example. This will take a couple of seconds to fire up.
The Binder service is a free service, which now installs all the dependencies and the code. Note that these dependencies are not installed on my computer. This is running in the browser. Anybody can visit this page with a browser. Here I can revisit all the steps. I can even try to modify them and see what happens if I change some parameters so this is really a reproducible and reusable data visualisation.

Another example: This was a Stanford Activity Inequality Study, where they have been studying phone tracking data. How much people move, and try to correlate movement with the daily activity

with cell phone tracking, in many different countries. What is interesting here is that, again, all the data, but also the data visualisation is available on GitHub. You can try to re rerun it.
Here is a collection of many more examples of interesting Jupyter notebooks for all different academic disciplines.

Python is not the only language popular in data visualisation. The other very popular programming language is R. R has something called R Markdown. It is the same idea as the Jupyter notebook, where we can have text and code all in the same place, and it can produce plots.
Just to show you how an R Markdown really looks, I will now compare the data visualisation book by Claus Wilke, which has been mentioned in a different video in the series. I will compare the R Markdown with the actual chapter in the book. You can see what the source is and the result.
So in the left panel is R Markdown. There is text and there is code. And on the right side is the corresponding text. What I see on the right-hand side are then the generated plots and now anybody can go in and can rerun this and can modify this.

This really nice figure is a summary of the challenges many students and researchers face today when they do data visualisation. It does not end with collecting data and collecting experiments. Here on the top left is a student who has done their research work. Now she would like to make the data visualisation reproducible, so that anybody can verify and reuse her plots. One popular way is then to use Jupyter notebooks or R Markdown to make the code and the plots publicly available, on a publicly hosted repository, such as GitHub or Bitbucket or GitLab. Then one can use the Binder service to make the notebook dynamically available, for anybody to rerun. One can even take it one step further. That would be to get a digital object identifier for the notebook to make it citable and to preserve it.

Here I have two examples, where I run a Jupyter notebook and a R Markdown (R studio) notebook, using the Binder service directly in the cloud. I will open these examples. I put them both on GitHub. Both have documented dependencies. I can now launch both through the Binder service. I will just start them, because both will take a couple of seconds to install all the dependencies, and then I will show you how these looks.
The dependencies are documented in these files. This example uses two libraries and I have documented not only the libraries, but also the different versions. I have done something similar for the R example.
My Jupyter notebook is already ready to be run. I can start it up here. I can run all the cells and create the plots for this example. It doesn't matter what we are plotting here. I'm not going into any details about how Jupyter or Python works.
Let's have a look at the RStudio/R Markdown example: Also this site launches through Binder. Those of you who have already used R Markdown and RStudio might be familiar with this interface. I can open up the R Markdown notebook and I can run everything. And here are some example plots. What is remarkable here is that anybody can run this. I could attach this notebook as supporting information to my publication, to make it really reproducible.

Finally, I would like to summarise that there is a progression. I have mentioned two languages, Python and R. I recommend learning the very basics in one of the two. Both are really good languages for data visualisation and statistics. Pick one of the two. Take the one that is maybe more popular in your academic community. It can also be a good idea to start learning right away, inside a notebook. For Python start with Jupyter. For R start with R Markdown and R Studio. Then try Binder later. Even later, learn how to get a digital object identifier for your Binder. Now your plotting recipe can be cited, is reproducible and preserved. This will take time, but that is OK. I believe that this

effort will really pay off. To make your plots reproducible, not only for other people but also for your future self.

Thanks so much for watching!