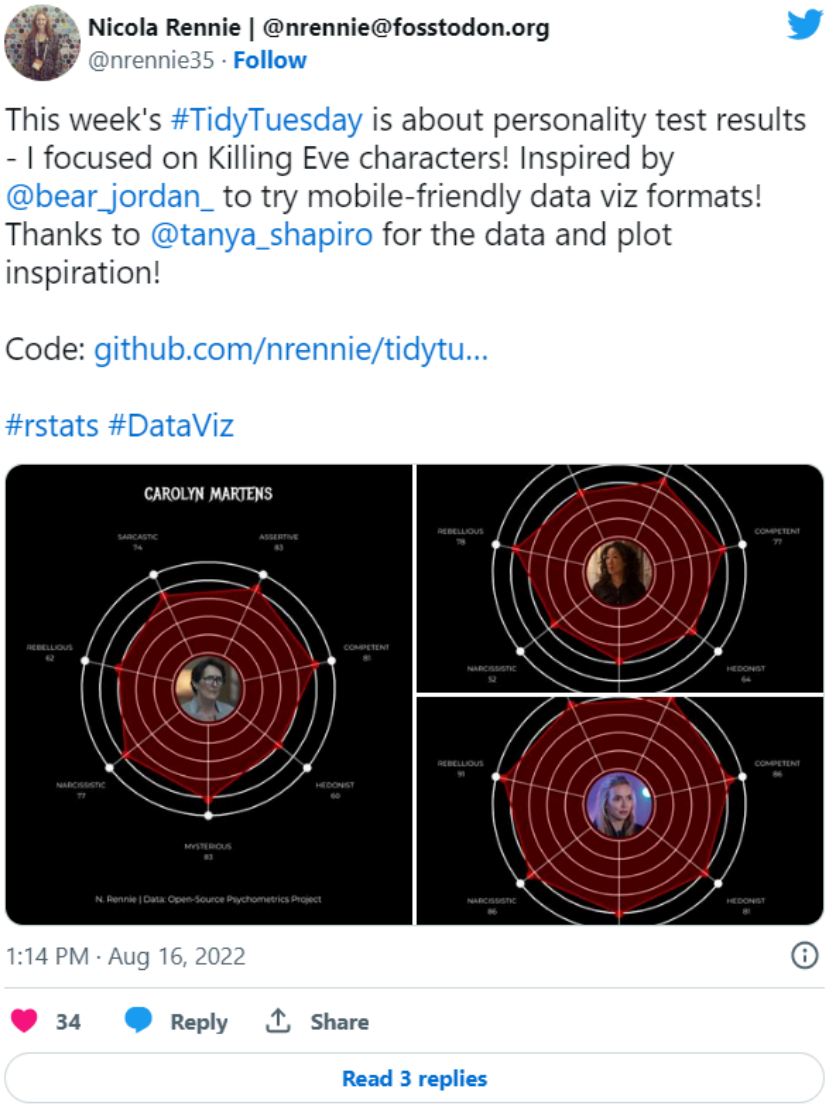
Designing #TidyTuesday visualisations for mobile (with Quarto)

If, like me, you mostly scroll through Twitter on your phone, you might want to consider designing your content specifically aimed at people who look at Twitter on their phone. Here's how to do it in R, with a little help from Quarto.

August 16, 2022

I've been contributing to #TidyTuesday challenges for about two years now, and my aim has always just been to play with new things in R. (For those of you unfamiliar with #TidyTuesday, it's a weekly data project aimed at helping people develop their data wrangling and visualisation skills.) For week 31, Bear Jordan created a very nice bar chart of the habitats of Oregon Spotted Frogs. One of the reasons this particular graphic caught my eye is that it was designed specifically for mobile viewing. And that got me thinking…

Most of the time, people (if they're anything like me) aren't viewing #TidyTuesday contributions on a laptop screen, they see them as they scroll through Twitter on their phone. Yet despite having made hundreds of #TidyTuesday visualisations, I've never intentionally thought about the way it would appear on a phone screen. And I'm guilty of that in other R development work - like Shiny apps. So this week, I decided to do something a little bit different - design my #TidyTuesday data visualisation specifically for viewing on mobile.



What does mobile-friendly visualisation mean?

In this context, I'm specifically talking about static images. Interactive dashboards and apps should also be designed for mobile, but that's a bigger topic for another day. For me, making a static data visualisation mobile friendly basically means two things:

- The aspect ratio of the image should be approximately 1:2 so that when someone clicks on the image to view it fullscreen, it takes up most of the screen.
- The image should be clear and easy to read on a smaller screen. Basically, there shouldn't be any need to zoom in or scroll to see the plot clearly.

Although these two things might seem straightforward, they brought up a couple of interesting problems. When I'm developing visualisations in R (or any other tool), I usually want to be able to do two things:

- Quickly preview plots as I'm iterating changes
- Save a high resolution version of the final image

Here's where the problems start…

Problem 1: You need to set a specific size of plot

As I mentioned earlier, a key part of designing for mobile is to choose a relevant aspect ratio for your plot. The easiest way to preview images in RStudio is through the **Plots** pane (usually found on the bottom right). Although you can change the size of the plot pane easily, you can't specify a height and width.

Solution: open a new graphics window.

You can open a new graphics window using the `dev.new()` function, which allows you to specify the width and height of the new window:

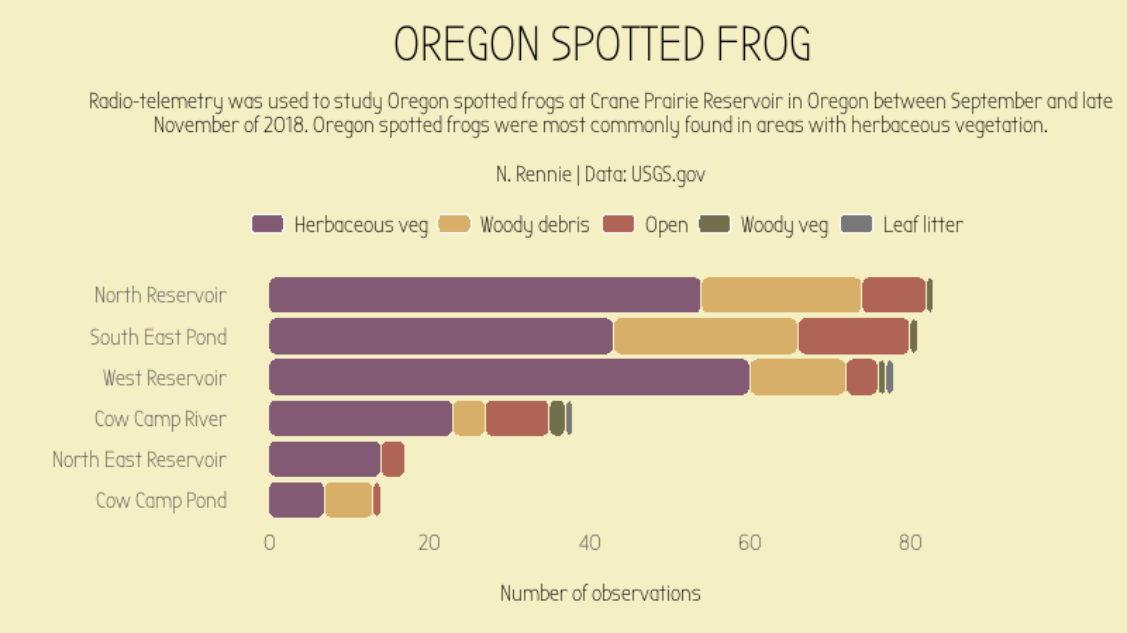
Copy

```
1 dev.new(width=1080, height=2160, unit="px", noRStudioGD = TRUE)
```

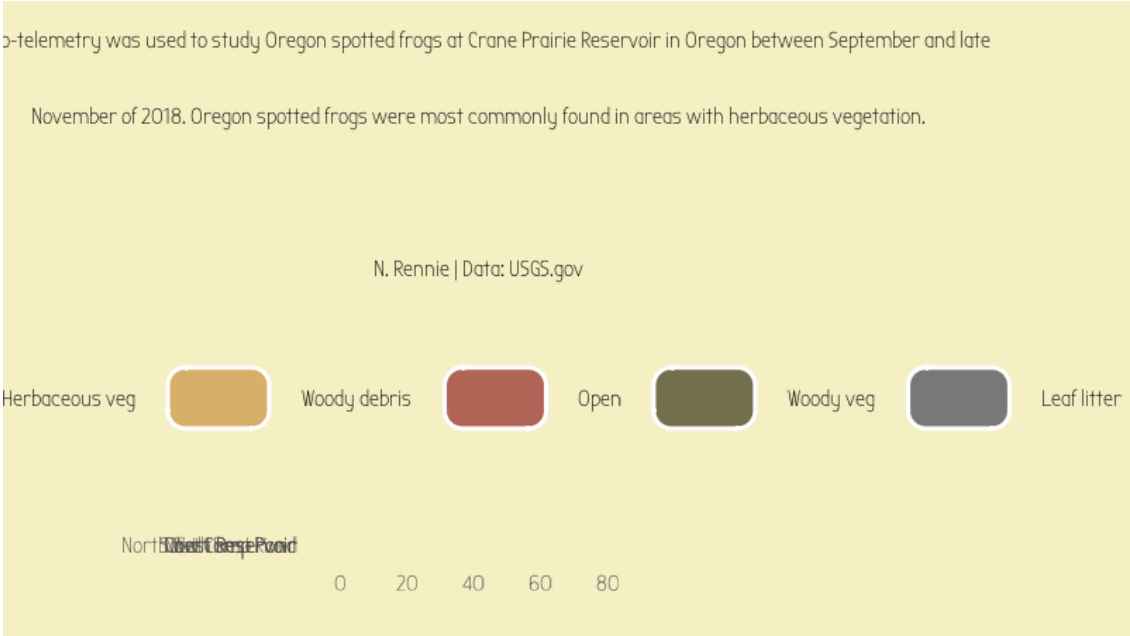
Here, I've specified the width and height in pixels. By setting `noRStudioGD = TRUE`, any new plots appear in the new graphics window rather than the RStudio graphics device. Other functions such as `windows()`, `x11()`, or `png()` from the {ragg} package can do similar things.

Problem 2: You need to save a high resolution image (that matches what you seen on screen)

Have you ever spent ages tinkering with a plot you're previewing in RStudio…



… and then used `ggsave()` to save a higher resolution image, and ended up with something like this:



If you're previewing plots in the RStudio graphics, or in one of the other options I suggested as a solution to problem 1 above, then you're (usually) looking at an image with a resolution of 96 dpi (dots per inch). In contrast, `ggsave()` uses 300 dpi by default - it's what makes the saved images have a higher resolution. Unfortunately, because we're previewing and saving with different dpi, our plots can look completely different after we save the final version. Although you can simply use `ggsave()` from the start, and repeatedly save over your plot, then open the file to preview it - I find that quite a tedious process and wanted something a little bit more automated.

P.S. If you're looking for a longer explanation of how dpi, pixels, and absolute measurements all fit together (and how they affect elements like font size), I'd recommend Christophe Nicault's blog post on the topic.

Solution: Quarto (or RMarkdown).

I'll preface this section with the fact that I already like Quarto a lot. If you've never heard of Quarto - think of it as the next-generation of RMarkdown with support for multiple programming languages and multiple output formats. And yes, everything in this solution also works with RMarkdown if you're not ready to make the change to Quarto.

Quarto (or RMarkdown) probably wasn't designed to help you make png or jpg files of static visualisations in a specific size, but it's a really nice feature that comes out naturally. Instead of running the code to generate your plot in an R script, run it in a code block in a Quarto (.qmd) or RMarkdown (.Rmd) file instead. The code block options for your code might look something like this:

Copy

```
```{r}
#| dpi: 300
#| fig.height: 7.2
#| fig.width: 3.6
#| dev: "png"
#| echo: false
#| warning: false
#| message: false
```
```

For each code block, you can specify the dpi, the height and width of the plot, and the file type. Here, I've set the plot height to be 7.2 inches (`fig.height` and `fig.width` are always given in inches), which with a dpi of 300, gives a height of 2160 pixels (the same as the `dev.new()` method above). To make it easier to simply preview your plot, rather than also seeing your code, you can hide your code and any messages or warnings by setting `echo`, `warning`, and `message` to `false`.

When you render your Quarto or RMarkdown document, it should appear in the **Viewer** tab in the bottom right corner of RStudio. The image that is rendered here will have the dpi you specified rather than 96 dpi as in the **Plot** tab. The plot will stretch to fit the pane width, but it will look *correct* in terms of the way the e.g. fonts appear. If you haven't set your Quarto document to be `self-contained`, then the images have also already been saved for you - probably in a folder called `documentname_files/figure-html/`. Otherwise, either right click on the image and save, or use `ggsave()` with the same dpi setting.

Another bonus: if you *run current chunk* in Quarto a preview of the plot will appear at 96 dpi. However, the plot is resized so that the preview of the current chunk looks the same as the high resolution version.

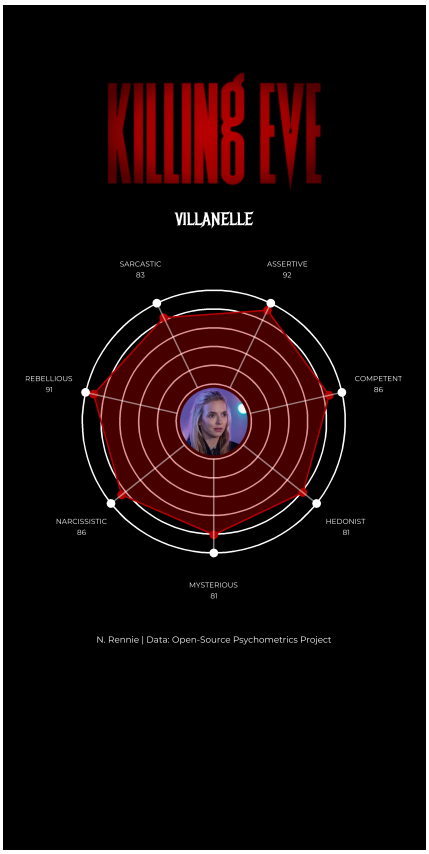
> Note: an alternative solution to both Problem 1 and 2 is to use {camcorder}, an R package to track and automatically save graphics generated with {ggplot2}. You can set the dpi, height, and width of your plot and it will appear in the Viewer pane of RStudio.

Copy

```
library(camcorder)
gg_record(
  dir = "recording",
  device = "png",
  width = 2,
  height = 4,
  units = "in",
  dpi = 300
)
```

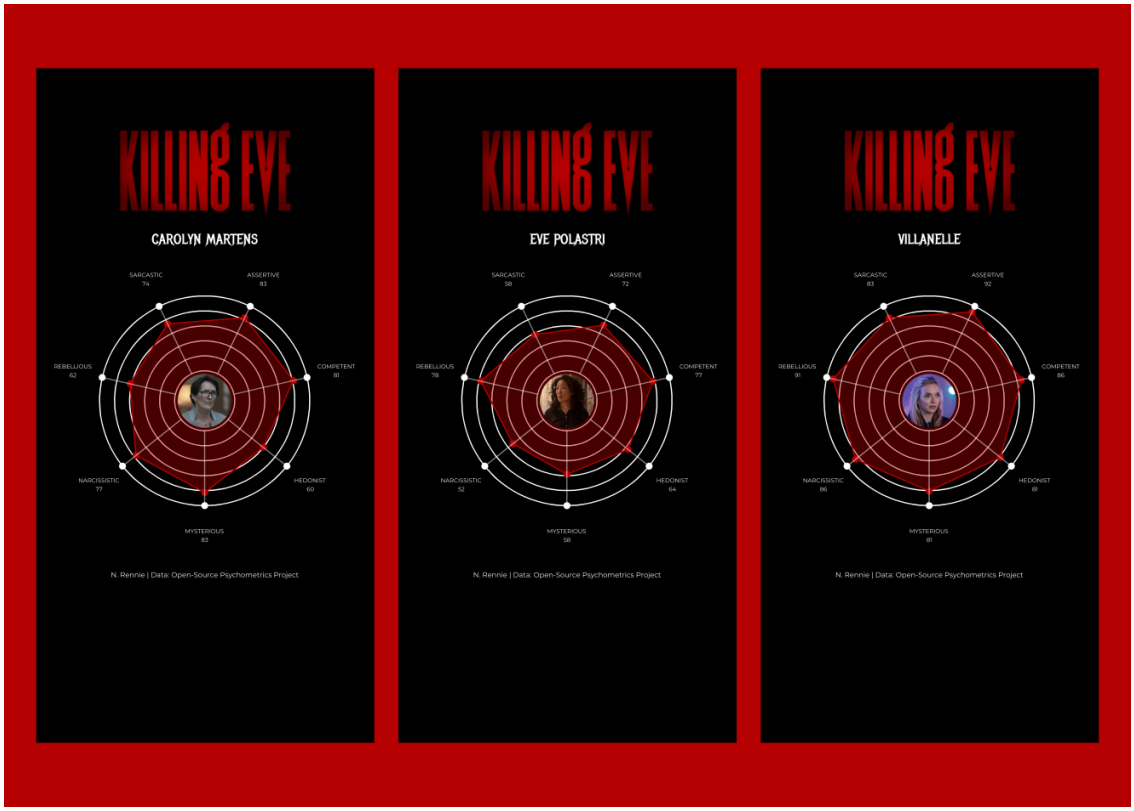
Problem 3: You have a lot less space to work with!

For me, the whole point of creating a visualisation specifically for viewing on mobile, is that you don't have to zoom in to see what the plot contains. This means your design needs to be reasonably minimal - I'd suggest a maximum of one plot, and maybe a (very short) paragraph of text. This felt a lot less than I'd normally put into a data visualisation. Usually, I'd have multiple plots, a paragraph giving context, some logos, annotations, …



Solution: Create more than one image.

Don't make plots with more in them, make more plots! For the example from this week's #TidyTuesday, I created three separate images, one for each of the three main characters in Killing Eve. Normally, I would use `facet_wrap()` here to automatically create a plot for each character and arrange them in a single row or column. But three plots would have been too much information, with plots too small, for a mobile screen. Instead, three separate images, each with the same theming and styling to create a gallery effect, works much better.



You can find my code for this week's #TidyTuesday on GitHub. Any other tips for creating data visualisations for mobile?

For attribution, please cite this work as:

> **Designing #TidyTuesday visualisations for mobile (with Quarto)**.
> Nicola Rennie. August 16, 2022.
> nrennie.rbind.io/blog/designing-tidytuesday-visualisations-for-mobile-with-quarto

← Mapping a marathon with {rStrava} Introducing {PrettyCols} →
