WeBIPP Poster Supplement

Jimmy Oh

Department of Statistics
University of Auckland
joh024@aucklanduni.ac.nz

1 More on WeBIPP

WeBIPP is a web-based interactive tool for building statistical graphics from scratch without having to write code. Conceptually, this is similar to the idea proposed by Bret Victor in early 2013[1], with Lyra[2] sharing many similarities to this idea.

WeBIPP, however, was developed independently and thus has a crucial difference in fundamental philosophy - WeBIPP does not shy away from code. Rather than focusing solely on the GUI-side of things, WeBIPP is a bridging solution, trying to take the best of both using GUIs, and writing code. WeBIPP is not only a tool for interactively producing graphics, it is a tool for interactively writing code. This has many advantages, including:

- All actions recorded and fully reproducible
- Ability to tweak code as needed to go beyond the interface
- Being able to generalise your actions and sharing this generalisation as an addon

Being able to generalise your actions is termed *Functionising Actions*, and works by turning your actions (which was writing WeBIPP code in the background) into a function. This is a powerful tool not only for sharing complete graphics, but also in re-using parts of your graphic, both in the same graphic, or elsewhere.

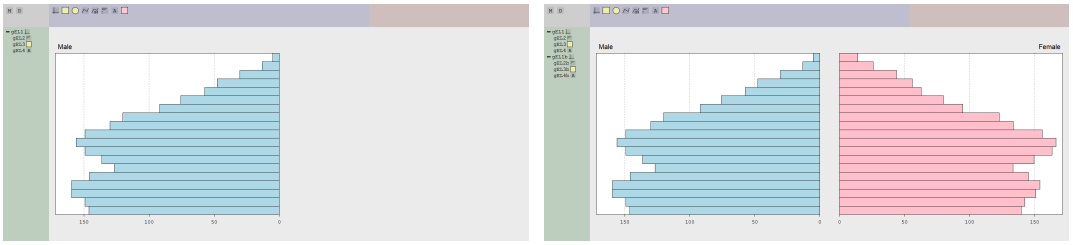

Figure 1: When creating a population pyramid, one only needs to create one side of the pyramid from scratch. The other side can be created by functionising the first half.

Also mentioned briefly in the poster is that almost everything in WeBIPP is an addon, allowing all manner of things, including the layout, the interfaces used for assigning attributes, and much more, to be altered through addons. The advantage of this is obvious, especially to users of R who benefit from the many packages R has. It enables great extension and customisation of WeBIPP to suit almost every need.

To facilitate this WeBIPP is extremely generalised. While this also has its advantages (mainly in being able to easily include new addons), it also means that WeBIPP's core system doesn't *understand* what it's doing. The implications of this are many, and range from a user interface that is perhaps, not as easy-to-use as a more restrictive software, to code that is less optimised as it must handle a very general case.

For more on WeBIPP, including a live version of WeBIPP that you can play with, and tutorials to replicate the examples from the poster, visit:
https://www.stat.auckland.ac.nz/~joh024/Research/WeBIPP/

[1] http://vimeo.com/66085662
[2] http://idl.cs.washington.edu/projects/lyra/

2 Using R with WeBIPP

Compared to the vast data manipulation capabilities of R, JavaScript is sorely lacking. While basic manipulation facilities will be added to WeBIPP in time, for now it only accepts JSON data in a `data.frame`-like format. Thus it is recommended for data to be sourced first into R (which can of course read from a variety of data formats) then exported to JSON.

2.1 Exporting to JSON directly

First step is to get a package that will allow exports to JSON. I use `rjson` for this purpose.

Second step is to get the data you want into a `data.frame`. The `rjson` package can accept a few different formats, but a `data.frame` is recommended for use with WeBIPP as it will ensure each of your variables will be of the same length.

Third step is to export the JSON to a file, which can then be loaded by WeBIPP. Examples follow:

```
library(rjson)
## Exporting cars data from R
## cars is already in a data.frame, so it's very simple
writeLines(toJSON(cars), "cars.json")

## Exporting Nile data from R
## Nile is a Time Series, and so cannot be exported directly
## We must first convert it to a data.frame
Nile.df = data.frame(Time = time(Nile), Nile)
writeLines(toJSON(Nile.df), "Nile.json")

## Exporting mother.tongue data
## Data available from the WeBIPP homepage
## mother.tongue is a named vector, so must be converted
mother.tongue.df = data.frame(lang = names(mother.tongue),
                              speakers = mother.tongue)
writeLines(toJSON(mother.tongue.df), "mothertongue.json")
```

2.2 Using integrated R terminal

It is also possible to make use of an integrated R terminal to connect R to WeBIPP, rather than exporting to a JSON file which is then read in by WeBIPP.

The current implementation makes use of the `shiny` package. Download the 'shiny App' (a folder containing some `.R` files used by the `shiny` package) *wbipR* from the WeBIPP homepage, then source *wbipR.R* to R.

```
library(shiny)
library(rjson)

## Ensure the working directory is the directory
## that contains the wbipR folder
source("wbipR.R")
runApp("wbipR")
```

This will open your default browser (refer to `shiny` documentation on how to change this, if required) and display WeBIPP, along with a terminal that grants full access to your R session. All inputs into this terminal are evaluated in the Global Environment of your R session with full permission.

The command `sendTOwbip` is a special command to send the given data to WeBIPP. For example `sendTOwbip(datacars = cars)` being run from the terminal will result in a new dataset called `datacars` being added to WeBIPP, which will contain the cars data.

Various ways to make this connection between R and WeBIPP more productive and intuitive are being worked on, and any suggestions are welcome.