A hack to plot images as points with ggplot2

Adam Lenart

Recently, while preparing for a conference, I wanted to create a scatter plot of nations with respect to their health care spending and life expectancy. As you can see below, the result was quite terrible. There are many countries that spend about the same amount of money (per capita) on health care and share a similar life expectancy at birth. There is no rotation of the country names that would save the legibility of the plot. I tried two- or three-letter country codes but the plot still looked terrible.

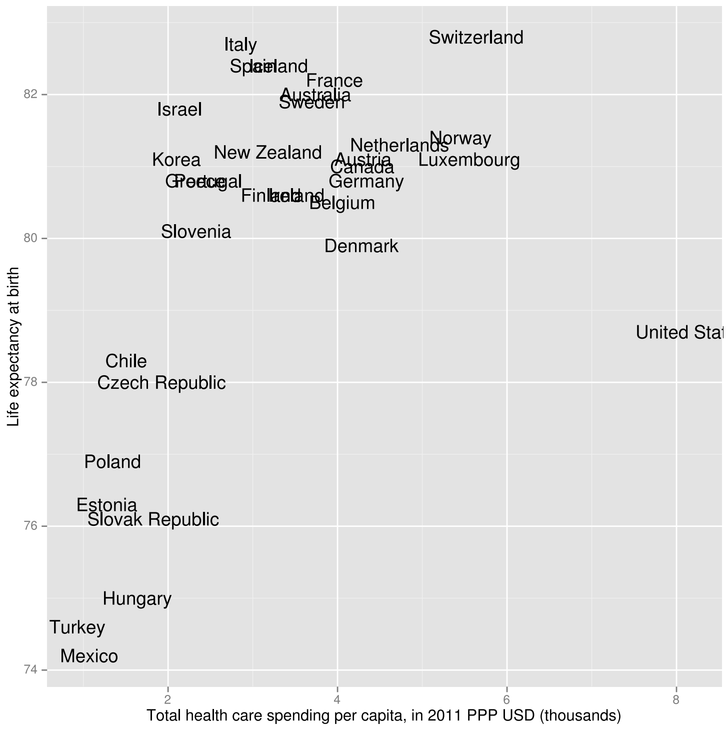

Figure 1: Nations plotted as flags

However, if we substituted the country names by their flags, it would better separate the countries and the plot would be more colorful as well. I was also counting on an audience that could identify the individual flags.

So how do we go about it?

- We need to have the flags as image files

Actually, this step was the most time consuming. After visiting many pages, I found a collection of svg flags. I can't recall where I found them right now but if I remember (ie., find it in the browser history), I will place a link here.

- Convert them into a suitable format for R to read in

The grImport package requires postscript files to import into R. I used Inkscape to convert the svg files into postscripts.

- Read the flags into an R object

The PostScriptTrace() function reads the ps files in and produces an XML document that can be read into R. Well, now that didn't work out of the box on my office Windows computer. For some reason, it didn't find my correct Ghostscript installation (I had to solve this a year ago for the first time and I don't really remember what exactly caused the problem). So what I did was to simply replace lines 15-22 of the PostScriptTrace()

```
            gsexe <- Sys.getenv("GSC")
        if (is.null(gsexe) || !nzchar(gsexe)) {
            poss <- Sys.which(c("gswin64c.exe", "gswin32c.exe"))
            poss <- poss[nzchar(poss)]
            gsexe <- if (length(poss))
                poss
            else "gswin32c.exe"
        }
```

by

```
 gsexe <- "gswin64c.exe"
```

and saved the modified `PostScriptTrace()` under the name of `pst()`. In an earlier version of `grImport`, I think `gsexe` (above) was called `gscmd`.

```
require(stringr)
require(grImport)
files <-  dir("c:\\Users\\alenart\\munka\\HealthEfficiency\\oecdData\\Flags")
 # the svg files are saved in the same folder, so I need to select the ps files only
files <- files[str_detect(files,"ps")]
# dp is the name of my data frame which contains the country data, one row per country
for(i in 1:nrow(dp)) {
   file <- files[i]
   pst(paste("C:\\Users\\alenart\\munka\\HealthEfficiency\\oecdData\\Flags\\",file,sep=""),
       outfilename=paste("C:\\Users\\alenart\\munka\\HealthEfficiency\\oecdData\\Flags\\",
     file,".xml",sep=""))
}
```

- Create the plot and place the flags on it

First, let's look at the data:

```
> head(dp)
         Country Efficiency Spending Lexp
1      Australia  0.9911519 3.747024 82.0
2        Austria  0.9800356 4.306831 81.1
3        Belgium  0.9728885 4.061404 80.5
4         Canada  0.9788323 4.294845 81.0
5          Chile  1.0000000 1.511557 78.3
6 Czech Republic  0.9672695 1.929181 78.0
```

then read the xml documents in by

```
files <- dir("c:\\Users\\alenart\\munka\\HealthEfficiency\\oecdData\\Flags")
files <- files[str_detect(files,"xml")]
k <- 1
pl <- list()
for(file in files) {
    pl[[k]] <- readPicture(paste("C:\\Users\\alenart\\munka\\HealthEfficiency\\oecdData\\Flags\\"
               ,file,sep=""))
  k <- k+1
}
```

```
names(pl) <- levels(dp$C)
```

and create the plot by plotting the countries only as points on the scatter plot for the time being. Here as an extra, I added the best practice frontier of life expectancy "production" but this has nothing to do with plotting images in ggplot2.

```
gp <- ggplot(data=dp,aes(x=Spending,y=Lexp))
gp2 <- gp+geom_point()

# add line of efficiency (it is a convex set so a simple sort works to
# get the right coordinates)

xE <- sort(dp$S[which(dp$E==1)])
yE <- sort(dp$L[which(dp$E==1)])
xEu <- c(xE[1],xE,ggplot_build(gp2)$panel$ranges[[1]]$x.range[2])
yEu <- c(ggplot_build(gp2)$panel$ranges[[1]]$y.range[1],yE,tail(yE,1))
d2 <- data.frame(X=xEu,Y=yEu)
gp3 <- gp2+geom_line(data=d2,aes(x=X,y=Y),size=1)+
    scale_y_continuous("Life expectancy",
                       limits=ggplot_build(gp2)$panel$ranges[[1]]$y.range,expand =c(0,0))+
    scale_x_continuous(
        "Total health care spending per capita, in 2011 PPP USD (thousands)",expand=c(0,0),
                                       limits=ggplot_build(gp2)$panel$ranges[[1]]$x.range)+
     # US data
    geom_segment(x=8.175,xend=8.175,y=78.7,yend=82.8,linetype=2)+
    # difference between best practice and US
                    geom_text(x=7.8,y=80.75,label="4.1")+
    geom_text(x=7.3,y=83,label="Best practice frontier")
```

Finally, the interesting part comes but I am a little embarassed about it. I don't know the inner intricacies of ggplot2 well enough to tweak `geom_point()` to plot the flags directly. Instead of it, as you will see, I will only superimpose the flags on the points.

```
pdf("flags.pdf",height=8,width=8)
gp3
seekViewport("panel.3-4-3-4") # from grid.ls()
xsource <- range(ggplot_build(gp3)$panel$ranges[[1]]$x.major_source)
xplot <- range(ggplot_build(gp3)$panel$ranges[[1]]$x.major)
ysource <- range(ggplot_build(gp3)$panel$ranges[[1]]$y.major_source)
yplot <- range(ggplot_build(gp3)$panel$ranges[[1]]$y.major)
xunit <- diff(xplot)/diff(xsource)
yunit <-  diff(yplot)/diff(ysource)

for(i in 1:length(pl)) {
  ycoord <- yplot[1]+yunit*(dp$Lexp[i]-ysource[1])
  xcoord <- xplot[1]+xunit*(dp$Spending[i]-xsource[1])
  grid.symbols(pl[[i]],x=xcoord,y=ycoord,size=unit(1,"cm"))
}
dev.off()
```

There are two interesting functions above: `ggplot_build()` and `grid.ls()`. The latter, `grid.ls()` lists the grobs used in the current ggplot2 plot. From this list, we can select the one that we wish to modify. E.g., if we wanted to plot images on the axes, we could have selected one of the axis grobs instead of the main panel.

The other function, `ggplot_build()` returns all of the necessary information about the ggplot2 plot. We are now focusing on the ggplot2 coordinates as they differ from the original coordinates of the data. By

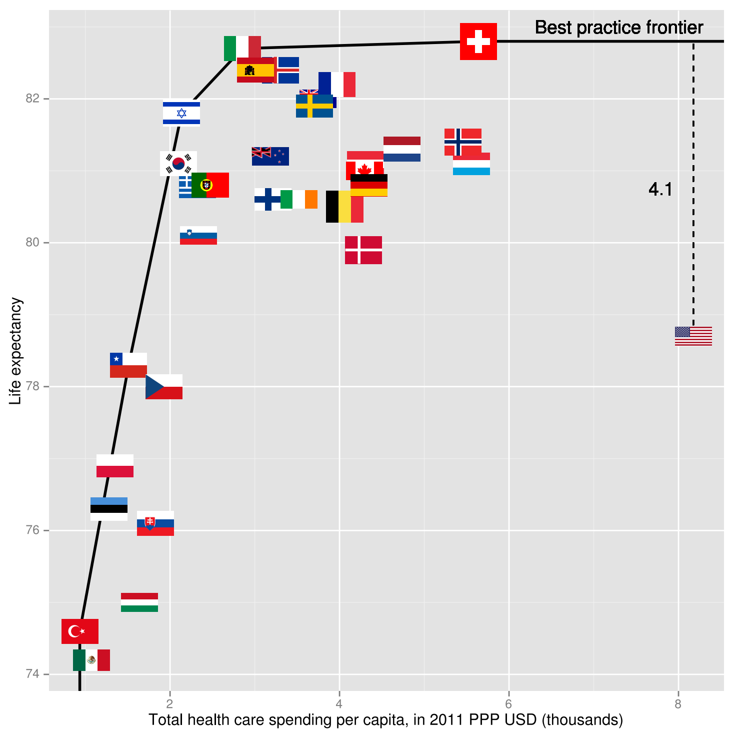

Figure 2: Nations plotted as flags

calculting a unit step on the axes, it is possible to translate our original coordinates into ggplot2 native ones. After that with `grid.symbols()` it becomes easy to overplot the flags on the points.