Was recently confronted to the following problem: creating hundreds of plots that could still be  
edited by our client. What this meant was that I needed to export the graphs in Excel or Powerpoint  
or some other such tool that was familiar to the client, and not export the plots directly to pdf or  
png as I would normally do. I still wanted to use R to do it though, because I could do what I always  
do to when I need to perform repetitive tasks such as producing hundreds of plots; map over a list  
of, say, countries, and make one plot per country.

So, after some online seaching, I found the {officer} package. This package allows you to put  
objects into Microsoft documents. For example, editable plots in a Powerpoint document. This is what  
I will show in this blog post.

Let’s start by loading the required packages:

library("tidyverse")

library("officer")

library("rvg")

.

You can download the data [here](https://github.com/rbind/b-rodrigues.github.com/blob/master/content/blog/clean_data.csv).

Let’s import and prepare it:

time\_use <- rio::import("clean\_data.csv")

time\_use <- time\_use %>%

filter(population %in% c("Male", "Female")) %>%

filter(activities %in% c("Personal care", "Sleep", "Eating",

"Employment", "Household and family care")) %>%

group\_by(day) %>%

nest()

I only kept two categories, “Male” and “Female” and 5 activities. Then I grouped by day and nested  
the data. This is how it looks like:

time\_use

## # A tibble: 3 x 2

## day data

##

## 1 Year 2014\_Monday til Friday

## 2 Year 2014\_Saturday

## 3 Year 2014\_Sunday

As shown, time\_use is a tibble with 2 columns, the first day contains the days, and the second  
data, is of type list, and each element of these lists are tibbles themselves. Let’s take a look  
inside one:

time\_use$data[1]

## [[1]]

## # A tibble: 10 x 4

## population activities time time\_in\_minutes

##

## 1 Male Personal care 11:00 660

## 2 Male Sleep 08:24 504

## 3 Male Eating 01:46 106

## 4 Male Employment 08:11 491

## 5 Male Household and family care 01:59 119

## 6 Female Personal care 11:15 675

## 7 Female Sleep 08:27 507

## 8 Female Eating 01:48 108

## 9 Female Employment 06:54 414

## 10 Female Household and family care 03:49 229

I can now create plots for each of the days with the following code:

my\_plots <- time\_use %>%

mutate(plots = map2(.y = day, .x = data, ~ggplot(data = .x) + theme\_minimal() +

geom\_col(aes(y = time\_in\_minutes, x = activities, fill = population),

position = "dodge") +

ggtitle(.y) +

ylab("Time in minutes") +

xlab("Activities")))

Let’s take a look at my\_plots:

my\_plots

## # A tibble: 3 x 3

## day data plots

##

## 1 Year 2014\_Monday til Friday

## 2 Year 2014\_Saturday

## 3 Year 2014\_Sunday

The last column, called plots is a list where each element is a plot! We can take a look at one:

my\_plots$plots[1]

## [[1]]

Now, this is where I could export these plots as pdfs or pngs. But this is not what I need. I need  
to export these plots as editable charts for Powerpoint. To do this for one image, I would do the  
following (as per {officer}’s documentation):

read\_pptx() %>%

add\_slide(layout = "Title and Content", master = "Office Theme") %>%

ph\_with\_vg(code = print(one\_plot), type = "body") %>%

print(target = path)

To map this over a list of arguments, I wrote a wrapper:

create\_pptx <- function(plot, path){

if(!file.exists(path)) {

out <- read\_pptx()

} else {

out <- read\_pptx(path)

}

out %>%

add\_slide(layout = "Title and Content", master = "Office Theme") %>%

ph\_with\_vg(code = print(plot), type = "body") %>%

print(target = path)

}

This function takes two arguments, plot and path. plot must be an plot object such as the ones  
contained inside the plots column of my\_plots tibble. path is the path of where I want to save  
the pptx.

The first lines check if the file exists, if yes, the slides get added to the existing file, if not  
a new pptx gets created. The rest of the code is very similar to the one from the documentation. Now,  
to create my pptx I simple need to map over the plots column and provide a path:

map(my\_plots$plots, create\_pptx, path = "C:\test.pptx")