In a [previous blog post](http://www.brodrigues.co/blog/2018-09-11-human_to_machine/) I have showed  
how you could use the {tidyxl} package to go from a human readable Excel Workbook to a tidy  
data set (or flat file, as they are also called). Some people then contributed their solutions,  
which is always something I really enjoy when it happens. This way, I also get to learn things!

@expersso proposed a solution without {tidyxl}:

Interesting data wrangling exercise in [#rstats](https://twitter.com/hashtag/rstats?src=hash&ref_src=twsrc%5Etfw).   
My solution (without using {tidyxl}): <https://t.co/VjuOoM82yX> <https://t.co/VsXFyowigu>

— Eric (@expersso) [September 12, 2018](https://twitter.com/expersso/status/1039894727808757761?ref_src=twsrc%5Etfw)

Ben Stenhaug also proposed a solution on his [github](https://github.com/stenhaug/shared/blob/master/tidyxl_bruno_blog.md)  
which is simpler than my code in a lot of ways!

Now, it would be too bad not to further analyze this data. I’ve been wanting to play around with  
the {flexdashboard} package for some time now, but never really got the opportunity to do so.  
The opportunity has now arrived. Using the cleaned data from the last post, I will further tweak  
it a little bit, and then produce a very simple dashboard using {flexdashboard}.

If you want to skip the rest of the blog post and go directly to the dashboard, just click [here](https://brodriguesco.shinyapps.io/time_use_luxembourg/).

To make the data useful, I need to convert the strings that represent the amount of time spent  
doing a task (for example “1:23”) to minutes. For this I use the {chron} package:

clean\_data <- clean\_data %>%

mutate(time\_in\_minutes = paste0(time, ":00")) %>% # I need to add ":00" for the seconds else it won't work

mutate(time\_in\_minutes =

chron::hours(chron::times(time\_in\_minutes)) \* 60 +

chron::minutes(chron::times(time\_in\_minutes)))

rio::export(clean\_data, "clean\_data.csv")

Now we’re ready to go! Below is the code to build the dashboard; if you want to try, you should  
copy and paste the code inside a Rmd document:

---

title: "Time Use Survey of Luxembourguish residents"

output:

flexdashboard::flex\_dashboard:

orientation: columns

vertical\_layout: fill

runtime: shiny

---

`` `{r setup, include=FALSE}

library(flexdashboard)

library(shiny)

library(tidyverse)

library(plotly)

library(ggthemes)

main\_categories <- c("Personal care",

"Employment",

"Study",

"Household and family care",

"Voluntary work and meetings",

"Social life and entertainment",

"Sports and outdoor activities",

"Hobbies and games",

"Media",

"Travel")

df <- read.csv("clean\_data.csv") %>%

rename(Population = population) %>%

rename(Activities = activities)

`` `

Inputs {.sidebar}

-----------------------------------------------------------------------

`` `{r}

selectInput(inputId = "activitiesName",

label = "Choose an activity",

choices = unique(df$Activities))

selectInput(inputId = "dayName",

label = "Choose a day",

choices = unique(df$day),

selected = "Year 2014\_Monday til Friday")

selectInput(inputId = "populationName",

label = "Choose a population",

choices = unique(df$Population),

multiple = TRUE, selected = c("Male", "Female"))

`` `

The Time Use Survey (TUS) aims to measure accurately how people allocate their time across different day-to-day activities. To this end, people are asked to keep records of all their activities in a time diary. For each activity, additional information is collected about whether or not the person was alone doing it or together with other persons, where did the activity take place, etc. The main studies on time use have been conducted to calculate indicators making possible comparative analysis of quality of life within the same population or between countries. International studies care more about specific activities such as work (unpaid or not), free time, leisure, personal care (including sleep), etc.

Source: http://statistiques.public.lu/en/surveys/espace-households/time-use/index.html

Layout based on https://jjallaire.shinyapps.io/shiny-biclust/

Row

-----------------------------------------------------------------------

### Minutes spent per day on certain activities

`` `{r}

dfInput <- reactive({

df %>% filter(Activities == input$activitiesName,

Population %in% input$populationName,

day %in% input$dayName)

})

dfInput2 <- reactive({

df %>% filter(Activities %in% main\_categories,

Population %in% input$populationName,

day %in% input$dayName)

})

renderPlotly({

df1 <- dfInput()

p1 <- ggplot(df1,

aes(x = Activities, y = time\_in\_minutes, fill = Population)) +

geom\_col(position = "dodge") +

theme\_minimal() +

xlab("Activities") +

ylab("Time in minutes") +

scale\_fill\_gdocs()

ggplotly(p1)})

`` `

Row

-----------------------------------------------------------------------

### Proportion of the day spent on main activities

`` `{r}

renderPlotly({

df2 <- dfInput2()

p2 <- ggplot(df2,

aes(x = Population, y = time\_in\_minutes, fill = Activities)) +

geom\_bar(stat="identity", position="fill") +

xlab("Proportion") +

ylab("Proportion") +

theme\_minimal() +

scale\_fill\_gdocs()

ggplotly(p2)

})

`` `

You will see that I have defined the following atomic vector:

main\_categories <- c("Personal care",

"Employment",

"Study",

"Household and family care",

"Voluntary work and meetings",

"Social life and entertainment",

"Sports and outdoor activities",

"Hobbies and games",

"Media",

"Travel")

If you go back to the raw Excel file, you will see that these main categories are then split into  
secondary activities. The first bar plot of the dashboard does not distinguish between the main and  
secondary activities, whereas the second barplot only considers the main activities. I could  
have added another column to the data that helped distinguish whether an activity was a main or secondary one,  
but I was lazy. The source code of the dashboard is very simple as it uses R Markdown. To have  
interactivity, I’ve used Shiny to dynamically filter the data, and built the plots with {ggplot2}.  
Finally, I’ve passed the plots to the ggplotly() function from the {plotly} package for some  
quick and easy javascript goodness!