**Package introduction:**  
Package myprettyreport helps to export ggplot2 graphs into a good-looking PDF file in a clear and easy way with a wide range of flexibility. It has a modular structure so the report elements can be combined in many ways.

**Installation:**

#install.packages("myprettyreport ")

**Quick overview:**

Functions of the package:  
start\_report() is the first mandatory function to generate the report.  
add\_cover\_page() function generates the cover page of the report.  
add\_new\_page() function adds a single report page to the report.  
add\_multiple\_page() function adds multiple report pages to the report.  
end\_report() function generates the final output and closes the process.

Quick example:

library(ggplot2)

sample\_plot <- ggplot(data = mtcars, mapping = aes(x = wt, y = mpg)) +

geom\_point() +

stat\_smooth(method = 'lm', color = "#f44242", fill = "#fd9068")

library(magick)

sample\_logo <- image\_read("http://myhappydata.com/img/1.png")

library(myprettyreport)

start\_report() %>%

add\_cover\_page(

logo = sample\_logo,

logo\_size = 0.3

) %>%

add\_new\_page(

plot = sample\_plot,

need\_header = TRUE,

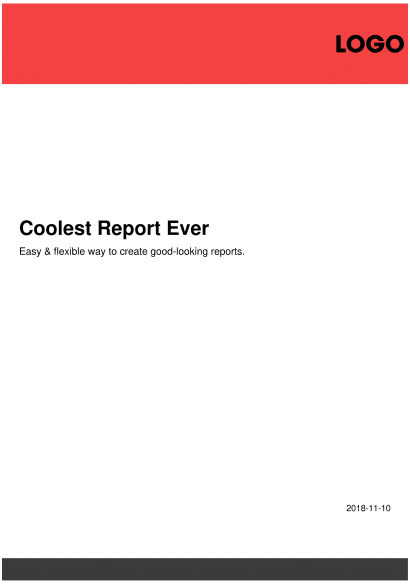
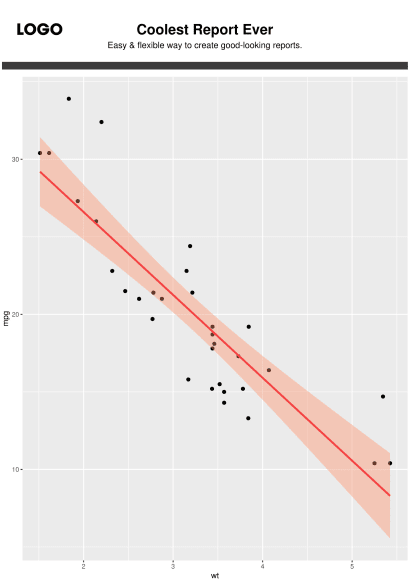
logo = sample\_logo,

logo\_size = 0.2

) %>%

end\_report()

Result:

**Basics**

A minimal example which generates a PDF file but it contains just a blank page:

start\_report() %>%

end\_report()

As we can see each elements can be added to each other with the magrittr::%>% (pipe) function.

Insert a new report page to a PDF file which creates a skeleton ggplot2 object as default.

start\_report() %>%

add\_new\_page() %>%

end\_report()

Now let’s create our first very own ggplot2 object and export it. The report syntax is almost the same except we have to add our ggplot2 object to a report as a plot parameter value.

library(ggplot2)

sample\_plot <- ggplot(data = mtcars, mapping = aes(x = wt, y = mpg)) +

geom\_point() +

stat\_smooth(method = 'lm', color = "#f44242", fill = "#fd9068")

start\_report() %>%

add\_new\_page(plot = sample\_plot) %>%

end\_report()

The result will be equivalent so far if we just use the ggplot2 built in export function:

ggsave("report\_output.pdf", width = 21, height = 29.7, units = "cm")

Although the result is the same it is easier to modify the report layout a little bit thanks to the package myprettyreport. We could create a footer and a header section like this:

start\_report() %>%

add\_new\_page(

plot = sample\_plot,

need\_header = TRUE,

need\_footer = TRUE,

footer\_text = "Copyright © Tamas Marko | 2018 | myhappydata.com") %>%

end\_report()

Also it could be possible to add a logo to a top left corner. All we need is to load a logo file into R and add it as a logo parameter. Optionally the size of the logo could be specified (valid values are between 0 and 1).

library(magick)

sample\_logo <- image\_read("http://myhappydata.com/img/1.png")

start\_report() %>%

add\_new\_page(

plot = sample\_plot,

need\_header = TRUE,

need\_footer = TRUE,

footer\_text = "Copyright © Tamas Marko | 2018 | myhappydata.com",

logo = sample\_logo,

logo\_size = 0.2) %>%

end\_report()

Currently 2 themes have been implemented yet. Now we are changing the theme from the default to a theme called flashy. In this case we have to modify the default values of the header and footer colors also because the default color of the text is white.

start\_report() %>%

add\_new\_page(

plot = sample\_plot,

need\_header = TRUE,

need\_footer = TRUE,

footer\_text = "Copyright © Tamas Marko | 2018 | myhappydata.com",

logo = sample\_logo,

logo\_size = 1,

theme = "flashy",

header\_color = "#f44242",

footer\_color = "#3d3c3c") %>%

end\_report()

We may notice that if we use this theme, the position of the logo will also change (now it is on the bottom left corner). So the size parameter of the logo will probably need to be reconfigured also.

The next step could be adding one more page to our report. In this case the preferred way is to use the add\_multiple\_page() function because it results more simplified syntax:

sample\_plot2 <- ggplot(airquality, aes(x=Temp)) +

geom\_histogram(binwidth=1, fill = "#f66f6f")

plot\_list <- list(sample\_plot, sample\_plot2)

start\_report() %>%

add\_multiple\_page(

plot = plot\_list,

need\_header = TRUE,

need\_footer = TRUE,

page\_number = rep("Copyright © Tamas Marko | 2018 | myhappydata.com", length(plot\_list)),

logo = sample\_logo,

logo\_size = 1,

theme = "flashy",

header\_color = "#f44242",

footer\_color = "#3d3c3c") %>%

end\_report()

Important to note that the add\_multiple\_page() function was executed (not the add\_new\_page()) and it has a page\_number parameter instead of footer\_text! So in this case we have to multiply the footer text value.

Now let’s add a cover page to our report. We have to insert the add\_cover\_page() function after the start\_report() function.

start\_report() %>%

add\_cover\_page() %>%

add\_multiple\_page(

plot = plot\_list,

need\_header = TRUE,

need\_footer = TRUE,

page\_number = rep("Copyright © Tamas Marko | 2018 | myhappydata.com", length(plot\_list)),

logo = sample\_logo,

logo\_size = 1,

theme = "flashy",

header\_color = "#f44242",

footer\_color = "#3d3c3c") %>%

end\_report()

It is also possible to change the theme of the cover page if we want the same “flashy” theme everywhere:

start\_report() %>%

add\_cover\_page(

theme = "flashy"

) %>%

add\_multiple\_page(

plot = plot\_list,

need\_header = TRUE,

need\_footer = TRUE,

page\_number = rep("Copyright © Tamas Marko | 2018 | myhappydata.com", length(plot\_list)),

logo = sample\_logo,

logo\_size = 1,

theme = "flashy",

header\_color = "#f44242",

footer\_color = "#3d3c3c") %>%

end\_report()

Now let’s add multiple plots to a single page:

start\_report() %>%

add\_new\_page(

plot = plot\_list,

plot\_hpos = c(1, 2),

plot\_vpos = c(1, 2),

plot\_area\_layout = grid::grid.layout(2, 2,

width = c(1, 1),

heigh = c(1, 1))

) %>%

end\_report()

In this case we have to specify the layout of the plot area with the grid::grid.layout() function. Also we have to determine the position of every single plot (plot\_hpos and plot\_vpos). For example if we create a 2×2 plot area plot\_hpos = c(1, 2) means that the first plot in the plot\_list will be in the left side of the page, and second one will be on the other side.

Optionally of course we could also use other libraries to reach the same goal. As I’ve mentioned already the result will be the same:

library(patchwork)

multiple\_plot\_one\_page <- sample\_plot + patchwork::plot\_spacer() +

patchwork::plot\_spacer() + sample\_plot2 +

patchwork::plot\_layout(ncol = 2, nrow = 2)

start\_report() %>%

add\_new\_page(

plot = multiple\_plot\_one\_page

) %>%

end\_report()

Finally create a report which has multiple pages and also multiple plots in every page:

plot\_list <- list(

list(sample\_plot, sample\_plot2),

list(sample\_plot2, sample\_plot)

)

start\_report() %>%

add\_multiple\_page(

plot = plot\_list,

plot\_hpos = c(1, 2),

plot\_vpos = c(1, 2),

plot\_area\_layout = grid::grid.layout(2, 2,

width = c(1, 1),

heigh = c(1, 1))

) %>%

end\_report()

So if we compare this syntax to the previous one we could see that the only difference is that we need to create sublists inside our list and every plot which has been defined inside a sublist will appear in the same report page.

**Advanced features**

**Export multiple ggplot2 objects into multiple files**

create\_multiple\_report <- function(db, xvar, yvar) {

sample\_plot <- ggplot(data = db, mapping = aes\_string(x = xvar, y = yvar)) +

geom\_point() +

stat\_smooth(method = 'lm', color = "#f44242", fill = "#fd9068")

start\_report(

filename = paste0(xvar,".pdf")

) %>%

add\_new\_page(plot = sample\_plot) %>%

end\_report()

}

lapply(colnames(mtcars)[5:7], create\_multiple\_report, db = mtcars, yvar = "mpg")

**Customize the report with extra grid parameters**

The package can allows us to pass extra grid parameters to our report to customize the final output. It can be useful if we need e.g. an additional text field. First of all we have to know the default layout structure of the report. If you haven’t heard about the basics of grid layouts it is recommended to go deeper into that topic at first. I share the structure of the layouts which was produced by the grid::grid.show.layout() function:

So e.g. if we want to insert an extra text field in the report page header when the theme is the basic, the syntax is the following:

rp\_e\_layout\_params = function() {

grid::grid.text("example text",

y = 0.7,

vp = grid::viewport(layout.pos.row = 1,

layout.pos.col = 1))

}

start\_report() %>%

add\_new\_page(

plot = sample\_plot,

need\_header = TRUE,

extra\_layout\_params = rp\_e\_layout\_params

) %>%

end\_report()