Commonly, the final product that a data scientist or a statistician generates is a report, usually in MS Word format. The [officer package](https://cran.r-project.org/web/packages/officer/index.html) enables generating such a report from within R. It also enables generating PowerPoint presentations, but this is beyond the scope of this post.

While the package has many great features,  using the package is not intuitive. The package manual covers no less than 80 pages. There are many functions that allow controlling many aspects of Word documents and PowerPoint presentation.

What I really need is to perform two tasks: inserting tables and figures into my Word document, and I also want to apply a standard format to all of my reports. I also need to add titles, empty lines and page breaks.

For these needs, I wrote a few functions that enable me to create standardized Word document reports using an intuitive syntax.

In addition to the officer package, the flextable package is also needed:

library(officer)

library(flextable)

The first function creates a new object that represents the Word document to be created. It simply wraps officer’s read\_docx() function, with the caveat that if an object with the same name already exists, it overrides it with a new “clean” object:

# create new word document

new.word.doc=function(){

my.doc=read\_docx()

return(my.doc)

}

The next two functions to add an empty line or a page break:

# add an empty line

add.empty.line=function(doc){

body\_add\_par(doc, " ")

return("empty line added")

}

#add page break

add.page.break=function(doc){

body\_add\_break(doc, pos="after")

return("page break added")

}

The next two functions are used to set the orientation of the next page to landscape, and then back to portrait:

# start landscape

start.landscape=function(doc){

doc=body\_end\_section\_continuous(doc)

return("landscape orientation started")

}

# end landscape

end.landscape=function(doc){

doc=body\_end\_section\_landscape(doc)

return("landscape orientation ended")

}

This function adds a title. I use the fp\_text() function to set the font size to 14, and the font type to bold. I also add an empty line after the title:

# add a title

add.title=function(doc, my.title){

my.prop=fp\_text(font.size = 14, bold = TRUE, font.family = "Times")

the.title=fpar(ftext(my.title, prop=my.prop))

body\_add\_fpar(doc, the.title)

body\_add\_par(doc, " ")

return("title added")

}

The next function is a little bit more complicated. It adds to the document a figure that already exists as an image file in the working directory. The parameters h and w indicate the height and width of image within the document, in inches. My usual values of choice are h=5 and w=5, so I set them as default. I align the figure to the center of the page.

# add an image, such as a jpg or png file

add.image=function(doc, image, h=5, w=5){

body\_add\_img(doc, src=image,

              height=h, width=w,

              style="centered")

return("image added")

}

The last function, add.table(), is the most complicated one. The table to be added is a data frame, and my function assumes that the variables in this data frame are either character, factor or numeric.

I have two parameters to set the number of decimals to present for each of the numeric variables. The first parameter receives the names of the numeric variables, the second contains the number of decimals for each of these variables, respectively.  Note that I do not check the inputs of these two parameters. I trust myself to insert the right input.

In addition, the function sets the format of the table (borders, fonts, and header formatting) according to my standard report style, using officer functions such as border\_outer(), bold() etc.

# add a data frame as a table

add.table=function(doc, tbl, col.keys=NULL, col.digits=NULL){

# create basic flextable

f.table=qflextable(tbl)

# set numbers of decimals for numeric variables, if specified

if(!is.null(col.keys)){

   for(j in 1:length(col.keys)){

     f.table=colformat\_num(x=f.table,

                           col\_keys=col.keys[j],

                           digits=col.digits[j])

   }

}

# set table borders

f.table=border\_outer(f.table, part="all",

                      border=fp\_border(color="black", width = 1))

f.table=border\_inner\_h(f.table, part="all",

                        border=fp\_border(color="black", width = 1))

f.table=border\_inner\_v(f.table, part="all",

                        border=fp\_border(color="black", width = 1))

# set fonts

f.table=font(f.table,  fontname = "Times", part = "all")

# also set the table's header font as bold

f.table=bold(f.table, part = "header")

# add the table to the document

flextable::body\_add\_flextable(doc,

value = f.table,

align = "left" )

return("table added")

}

Now we are all set to create the report:

# create an histogram and save it as a png file:

png(filename="histogram.png", width = 6, height = 6, units = 'in', res = 300)

hist(mtcars$wt)

dev.off()

# create a data frame that will become a table in my report

wide.table=mtcars[1:6, ]

wide.table$car=rownames(wide.table)

wide.table=wide.table[, c(12, 1:11)]

narrow.table=wide.table[, 1:4]

# create a new document object

doc=new.word.doc()

# add the report title and an empty line

add.title(doc, "My report")

add.empty.line(doc)

add.title(doc, "narrow table")

add.table(doc, narrow.table, col.keys=c("mpg", "disp"), col.digits=c(1,0))

add.page.break(doc)

# add the histogram with an apropriate title

add.title(doc, "Histogram - portrait")

add.image(doc, "histogram.png", h=3, w=3)

# set the orientation to lndscape

start.landscape(doc)

add.title(doc, "narrow table - landscape")

add.table(doc, narrow.table, col.keys=c("mpg", "disp"), col.digits=c(1,0))

add.empty.line(doc)

# add the wide table in landsacape page orientation

add.title(doc, "wide table in landscape orientation")

add.table(doc, wide.table, col.keys=c("mpg", "disp"), col.digits=c(1,0))

#set the orientation back to portrait

end.landscape(doc)

add.title(doc, "narrow table")

add.table(doc, narrow.table, col.keys=c("mpg", "disp"), col.digits=c(1,0))

# generate the Word document using the print function

print(doc, target="My report.docx")

Since I cannot upload a Word document to WordPress, here is a screen print of the report: