How to create a PDF code report in RStudio

Carlo Knotz

You need to submit commented *code reports* as PDFs with all of your three mandatory assignments.

These reports must contain:

- All of the code you wrote to produce the results you present in your assignment text.
- Brief (1-2 sentences) explanations of what a given part of your code does.

This document shows you how you can automatically create these reports with a single click in `RStudio`, and it provides a few tips for how to organize your code.

1 Step 1: Install the `tinytex` package (if you haven't already)

You should already have installed all the necessary software to create PDF reports when you went through Tutorial 0 – but just in case you did not: You need to install a package called `tinytex`. `tinytex` is really nothing more than a tool to install the `TinyTex` software automatically through *RStudio*, which you need to be able to create PDF code reports.[1]

You install `tinytex` just like the other packages:

```r
install.packages("tinytex")
```

Once that is complete, you run the following in your *Console* to install the `TinyTex` software:

```r
tinytex::install_tinytex()
```

The download may take a bit of time (and there'll be code gibberish).

[1] `TinyTex` is a small version of the `LaTeX` typesetting software on your computer, which `RStudio` then uses to create the code reports (see also https://yihui.org/tinytex/).

2 Step 2: Do your analysis and document your code

If you want to create a code report, you obviously need to have some code and results first. Step 1 is therefore to do your analysis in `RStudio` and create all the results you want to/need to present in your assignment.

To make everything as smooth as possible, use the code structure guide (provided on *Canvas*) to organize your code from the start. See also the example code report (on *Canvas*) to see how yours should ideally look like.

3 Step 3: Make sure everything works without problems

You can only create a report if your code runs from the start to the end without producing an error message ("breaking"). This means that there cannot be any typos, etc. in your code.

You can easily test if everything works by doing the following:

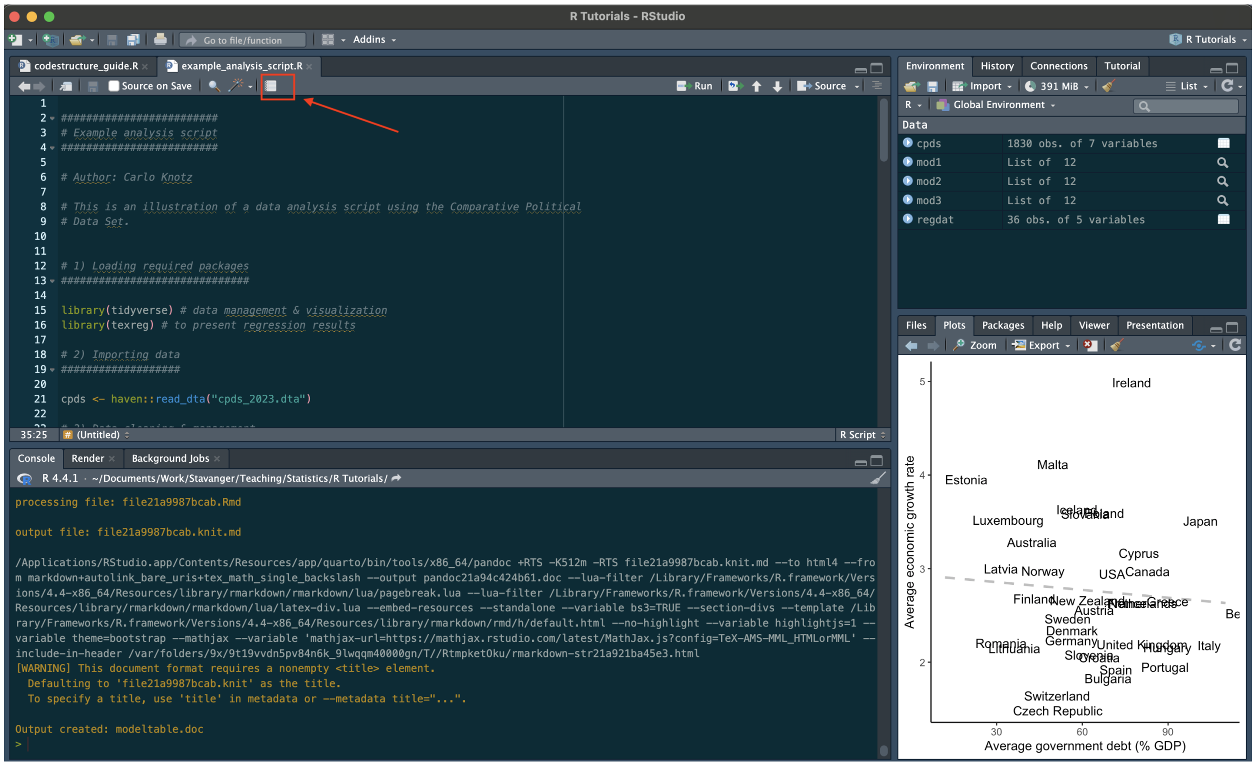
- Re-start `R` (in `RStudio`, go to *Session* in the menu at the top, then click on *Restart R*).
- Make sure that `R` does not load the previous results and data (see also Tutorial 0, section 4.1). Your *Environment* window should be empty.
- Select all the code in your scriptfile (right-click, *Select all*)
- Run the code (e.g., by clicking on *Run* in the menu bar at the top right of your scriptfile)

If everything works, then `R` will now execute all your code from the start to the end and reproduce all your results, one after the other. **Make sure that all the results (graphs, tables, statistical tests) are the exact same as those you present in your assignment.**

On the other hand, if there is an error in your code, `R` will stop and give you an error message. Obviously, you then need to fix this error ("de-bug" your code).

4 Step 4: Create the report

If you are sure that your code runs without errors or problems, then all you have to do is to click another little button as shown in the screenshot below:



```r

##########################
# Example analysis script
##########################

# Author: Carlo Knotz

# This is an illustration of a data analysis script using the Comparative Political
# Data Set.


# 1) Loading required packages
###############################

library(tidyverse) # data management & visualization
library(texreg) # to present regression results

# 2) Importing data
####################

cpds <- haven::read_dta("cpds_2023.dta")

```

Data	
cpds	1830 obs. of 7 variables
mod1	List of 12
mod2	List of 12
mod3	List of 12
regdat	36 obs. of 5 variables

```
processing file: file21a9987bcab.Rmd

output file: file21a9987bcab.knit.md

/Applications/RStudio.app/Contents/Resources/app/quarto/bin/tools/x86_64/pandoc +RTS -K512m -RTS file21a9987bcab.knit.md --to html4 --fro
m markdown+autolink_bare_uris+tex_math_single_backslash --output pandoc21a94c424b61.doc --lua-filter /Library/Frameworks/R.framework/Vers
ions/4.4-x86_64/Resources/library/rmarkdown/rmarkdown/lua/pagebreak.lua --lua-filter /Library/Frameworks/R.framework/Versions/4.4-x86_64/
Resources/library/rmarkdown/rmarkdown/lua/latex-div.lua --embed-resources --standalone --variable bs3=TRUE --section-divs --template /Lib
rary/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/rmarkdown/rmd/h/default.html --no-highlight --variable highlightjs=1 --
variable theme=bootstrap --mathjax --variable 'mathjax-url=https://mathjax.rstudio.com/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML' --
include-in-header /var/folders/9x/9t19vvdn5pv84n6k_9lwqqm40000gn/T//RtmpketOku/rmarkdown-str21a921ba45e3.html
[WARNING] This document format requires a nonempty <title> element.
  Defaulting to 'file21a9987bcab.knit' as the title.
  To specify a title, use 'title' in metadata or --metadata title="...".

Output created: modeltable.doc
>
```

Figure 1: Creating a code report

A small popup window asking you in which format you want to create the report should appear now – obviously, select PDF.

RStudio should then create the report and open it as a PDF window. It may also install a few additional packages when you do this for the very first time.

5 Step 5: Check the report

As before: Make sure that all the results (graphs, tables, statistical tests) are the exact same as those you present in your assignment.

It is fine if the report looks a bit messy as long as all the results are there and a reader can identify them. You have your main text to present the results in a neat and polished fashion.