Template using R, Sweave, knitr, and LaTeX from RStudio with .Rnw file

Matthew C. Ingram[1] Nakissa Jahanbani[1] Cesar Renteria[2]

[1] Department of Political Science
[2] Department of Public Affairs & Policy
Rockefeller College of Public Affairs & Policy

September 7, 2017

Abstract

Summary. Type your summary or abstract here, if desired. If there is none or if this part is not relevant, then omit.

1 Purpose of this document

This document offers you a template to improve your workflow in R using LaTeXand Sweave package.

2 Template in greater detail

This is the template for all assignments you produce for RPAD/RPOS 517. Please use this template for each assignment, and please submit all assignments as PDF documents.

There are three versions of the template:

1 one for Stata using MarkDoc and LaTeX(.do file);

2 one for R using Sweave and LaTeX(.Rnw)

3 one for R using knitr and markdown (.Rmd file);

This template is the second one: R with Sweave and LaTeX.

Each template offers a combination of three main tools, including a statistical package (Stata or R), a text processing and presentation system or language (LaTeX, markdown, and/or HTML), and a system for weaving together code from the statistical package and the text and formatting from the text processing and presentation system (MarkDoc, Sweave, or knitr).

What you choose will likely depend on your needs, and this may vary from one project to another. For instance, HTML may be a better choice for producing web pages and other web content, and markdown may be a better choice for producing short reports with little formatting. It may also be the case that one collaboration you have uses one set of tools and another team you work with uses a different set of tools.

In general, however, I recommend learning and using LaTeX; this is a powerful document preparation and typesetting system that will meet all of your needs.

3 Getting started

At this point, I assume the following:

- you have downloaded and installed R

- you have downloaded and installed RStudio
- you have downloaded and installed a .tex distribution and .tex editor (see Notes on LaTeX)

If all of the above is correct, open RStudio.

This .Rnw file using Sweave and LaTeXcan then be easily opened and edited in RStudio. The letters "nw" stand for "noweb", and ".nw" or "Rnw" are typical file extensions for Sweave files. For more on Sweave, see package document here.

RStudio will automatically call the version of R you have installed.

Once you have opened this .Rnw file in RStudio, you can run this .Rnw file by:

- clicking the "Compile PDF" button at the top of the screen (or typing CTRL-SHIFT-K on Windows or CMD-SHIFT-K on Mac)

This process of compiling (integrating all your code and text into a single automatic document) will produce a final PDF document in your working directory, but it will not open automatically; you need to go to directory and open the file; a .tex document will also be produced along the way; you can edit the original .Rmd document or the .tex document, though I recommed editing original .Rnw file so that the changes to the .tex file will not be lost.

Note: under Tools –> Global Options –> Sweave, you can switch between Sweave and knitr.

For compiling your .tex file (on the fly by RStudio or on your own) and any related figures, tables, or other output, you will need a .tex distribution and a .tex editor. There are several distribution and editors available, but I recommend the MiKTeX distribution and TeXstudio editor for Windows, and the MacTeX distribution and TeXshop editor for Mac.

To read a basic introduction to LaTeX, including suggestions for .tex distributions and editors, see

4 Quick intro to LaTeX

A few useful LaTeXcommands for writing are:

5 This is heading 1

This is a heading 1 without numbering

5.1 This is first subheading (heading 2)

This is second subheading (heading 2) without numbering

This is a text paragraph which requires no syntax.

This is a list of items:

- item 1
- item 2
- item 3

This is a numbered list:

(1) first item

(2) second item

(3) third item

5.2 Equations

This is a line introducing Equation 1 and Equation 2 below.

$$\alpha = \sqrt{\beta} \tag{1}$$

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{2}$$

You can also write equations inline like this $\alpha = \sqrt{\beta}$ or break other equations out with double dollar signs like this:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

5.3 Inline R code

Lastly, while most of your R code will be in code chunks, as shown below, you can also include short bits of R code inline. For example, the mean of 100 random numbers drawn from a uniform distribution between 1 and 100 is: 54.7367061874946.

6 Data Analysis Example

In the blocks of R code (called "code chunks") and text below (called "documentation chunks"), we illustrate several typical steps in the course of writing a paper, including:

(1) loading some data,

(2) describing the data, including a table of summary statistics,

(3) graphing the data,

(4) running a simple regression model,

(5) graphing the results.

Data are in a folder called "data", and all tables are exported to and then imported from a folder called "tables", and all figures are exported to and then imported from a folder called "figures".

7 Loading Data

In this section, we load our data.
We use the winter olympics dataset for Michael Bailey's statistics text, *Real Stats* (Oxford, 2016).
Output will not appear in final document if option "include=FALSE" is used. All R commands have been suppressed with global option above of "echo=FALSE".

A good practice is to name your objects with reference to the characteristics of the object. For example, variable name *data.df* using suffix *df.* can be used to indicate that object is a dataframe (df).

8 Exploring Data and Descriptive Statistics

Next, we can have take a quick look at the data set by just typing the object *data.df*. Or use other commands, such as head(), tail(), and summary(). You can control the summary for a single variable by adding the sympbol $ and after that, the name of the variable. For example summary(data.df$year).

R provides a wide range of more sophisticated packages for generating tables, including "stargazer". The output from this package is reported in Table 1.

If project involves working with a lot with variables, we might want to use a more practical route than, for example, "data.df$medals". We can use the attach/dettach commands to use only "year" as a reference

to the variable, rather than the former path. Attaching will get you into a more familiar environment when managing variables and datasets. If you are planning to use only one dataset in your project, you're completely safe. Otherwise, never forget to detach the data. Notice that if you generate a new variable, you need to detach and re-attach to see it.

Some people think that the attach function should never be used. Here are some reasons why.
1. every attach is a copy of the dataset in the memory. So, if you attach twice the same dataset and make any changes in it, you are not sure that the correct values are being used. Or, if you happen to work with two datasets that share some variable names, you're not sure what variable you are really calling.
2. Also, if you have 2 different data sets and they both have a variable with same name, e.g., 'y', or 'y-hat', or 'x1', then when you attach these data sets the variables look the same in R, and you cannot be sure which one is being called from memory.

**** Bottom line: use 'attach' very carefully ****
Here we import the table of summary statistics.

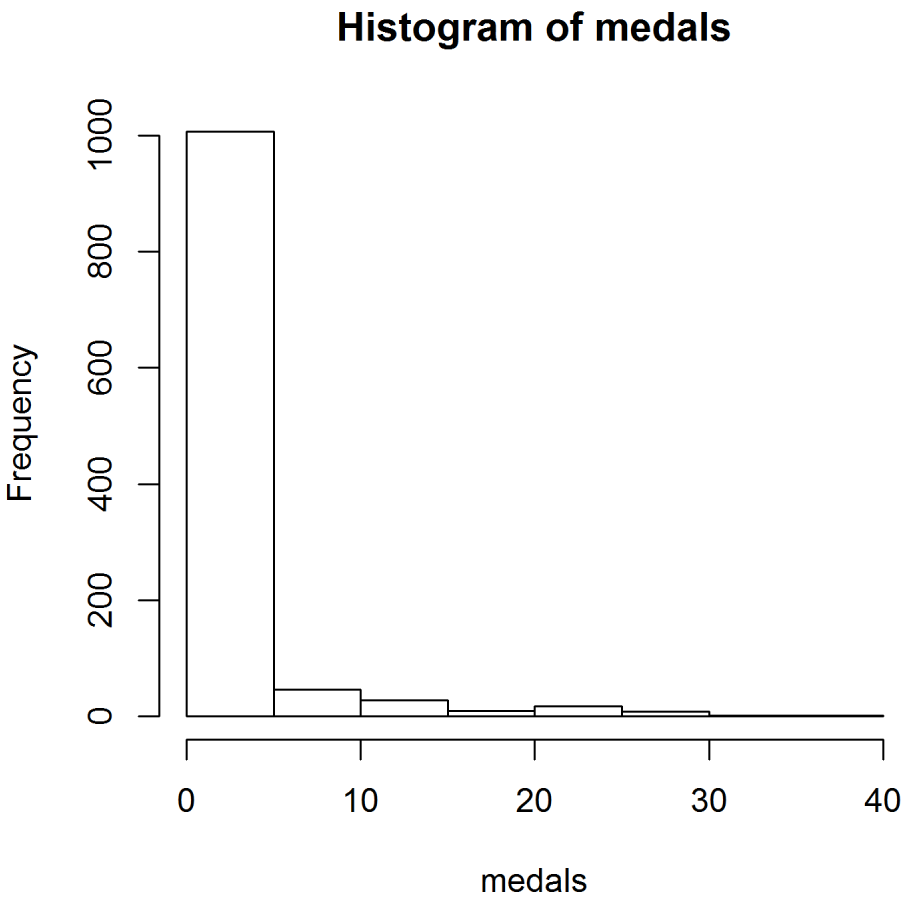
Table 1

Statistic	N	Mean	St. Dev.	Min	Max
ID	1,122	58.880	33.990	1	118
year	1,122	1,997.000	10.700	1,980	2,014
host	1,121	0.009	0.094	0	1
temp	1,100	53.950	19.330	3.900	90.500
precipitation	1,100	56.040	45.030	0.300	280.000
elevation	1,100	3,018.000	2,195.000	43	8,850
gold	1,122	0.587	1.926	0	14
silver	1,122	0.584	1.837	0	16
bronze	1,121	0.581	1.696	0	13
population	1,113	51.930	179.000	0.000	1,351.000
GDP	966	1.169	1.807	0.011	14.520
participate	1,122	0.601	0.490	0	1
medals	1,122	1.751	5.171	0	37
athletes	1,122	18.170	36.880	0	230
time	1,122	5.423	2.882	1	10

Now lets generate and add a figure that visualizes one of these variables.

Traditional command for plotting is *plot()*. However, there is an alternative, ggplot(), that substantially improves the look of your graph and is extremely flexible. You should start using ggplot as soon as possible.

The basic structure of *gplot()* is this. First, you identify the dataframe and the variables of interest within ggplot(). This command is similar to Stata's *twoway*, since it does nothing by itself, but sets the structure for a graphing command. Then, you have to add elements to the plot with the sign "+". This sign resembles the comma in Stata's graphing commands for adding options. The command *geom_point()* resembles *scatter* in Stata, because this is going to add the points to the graph; "labs" stands for labels.



9 Regression

Let's end this template with a regression analysis.

we need to call the command lm(), which stands for linear model. Within the command we first place the dependent variable "medals". Next, we separate the dependent variable from the independents with the ' ' symbol. Fourth, we type the independent variable(s) separated by the sign; in this case, only athletes, GDP, and temp. Finally, unless data are attached, we must either call both dataset and variable (e.g., data.df$medals) or define the source dataset; in this case, data.df.

We need to store the results in an object, so we can use them in post-estimations. To do that, run the same coding line, but assign it to a new object. Here, we call new object "m1" (model 1) and "m2" (model 2). Results are stored, but you cannot see them. To view the results, we must call or manipulate the "m1" object (or "m2").

Next, we output results cleanly in different formats.

In the code that generated this table (stargazer package), we labeled a table as "regresults", so we can call that table as Table 2.

Finally, let's graph the OLS results.

Then we can write some text and reference the figure by noting that Figure 1 and Figure 2 report the coefficients with 95% confidence intervals.

Or we could plot the two figures together as they appear in Figure 3 and Figure 4 of Figure 5.

Or try something different as in Figure 6.

Table 2: Results

	Dependent variable:	
	medals	
	(1)	(2)
temp		0.011**
		(0.005)
athletes	0.113***	0.113***
	(0.003)	(0.003)
GDP	0.123**	0.172***
	(0.057)	(0.054)
Constant	-0.477***	-1.225***
	(0.119)	(0.347)
Observations	966	956
R^2	0.685	0.707
Adjusted R^2	0.684	0.706
Residual Std. Error	2.989 (df = 963)	2.818 (df = 952)
F Statistic	1,046.000*** (df = 2; 963)	765.600*** (df = 3; 952)
Note:		*p<0.1; **p<0.05; ***p<0.01

DETACH. Consider this as roughly equivalent to Stata's 'clear' (not 'clear all'). Make detaching a strict practice or habig, or else avoid using attach/detach entirely. Remember that any attached data will be in R's search path, so any command that is executed will search through the attached data and apply action to a variable name.

Assuming you have detached your data, it will still be in memory. This is a nice feature of R compared to Stata. Even though we will load a new dataframe, the previous one (dt.olympics) will remain in memory and we can call it whenever we want. Just remember that you may have multiple data sets available if you save your current workspace for a particular project.

10 Conclusion

Finally, we would write our conclusion here.

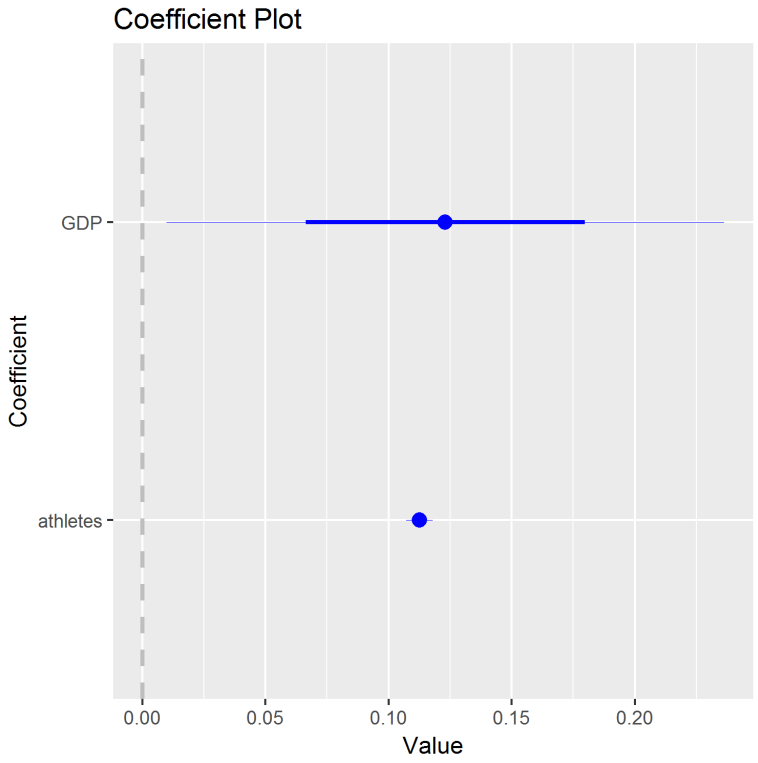


Figure 1: This graph shows the coefficients for the first regression.

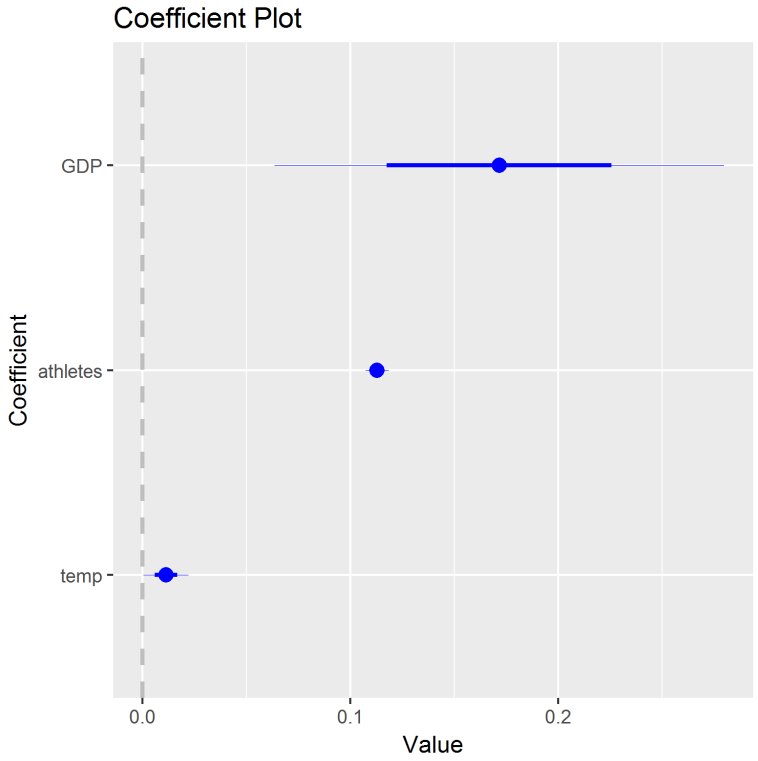
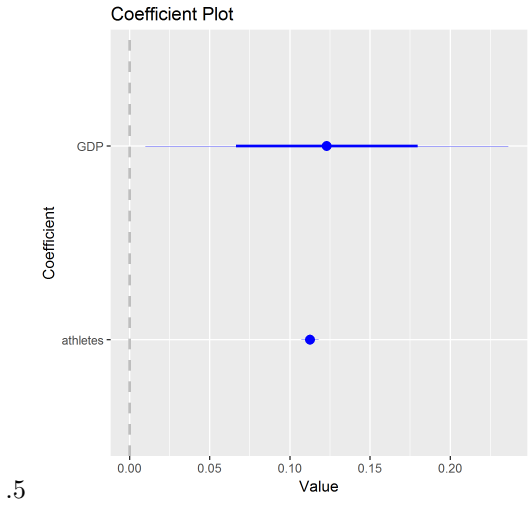
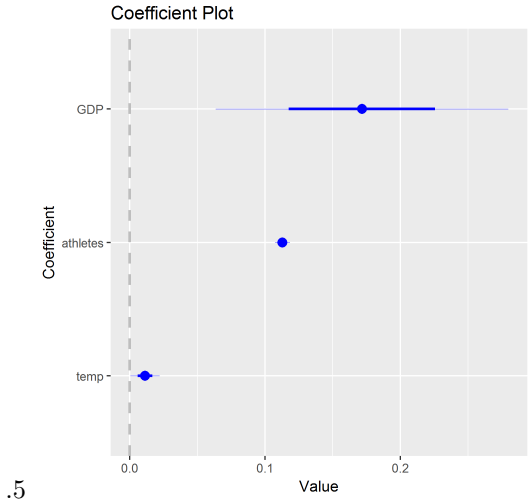


Figure 2: This graph shows the coefficients for the second regression.



.5

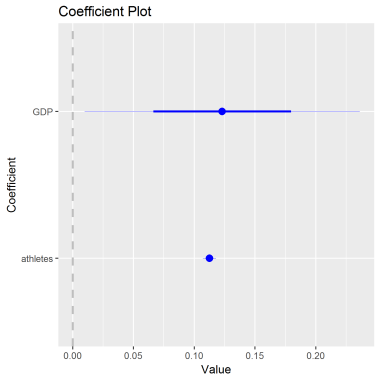
Figure 3: Model 1



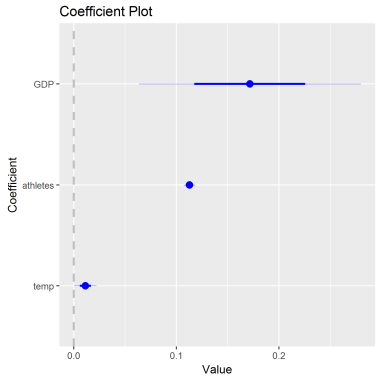
.5

Figure 4: Model 2

Figure 5: A figure with two subfigures



((a)) label 1



((b)) label 2

Figure 6: 2 Figures side by side