Module14: Tips for Final Report

Brian Steves

May 19, 2016

ESM 505/ESR 605 : Data Management, Spring 2016

==========

Setting Up Your Report

Git

Use Git and either Bitbucket or Github to store your project.

This will allow you better version control and a built in off site backup. It will also allow you to upload your data and a URL to your repository as your final report.

If you're worried about your data, add the data file names to your .gitignore file and they won't be included in your repository on GitHub or BitBucket. Just be sure to add the data to your D2L final report dropbox.

Use Relative Links

Be sure to use links relative to your projects working directory.

For example.. dat <- read.csv("data/mydata.csv") will retrieve the csv file from the "data" sub-directory of your project folder. When I attempt to reproduce your final report this should work just fine.

However, if you use absolute links, things are likely to not work as well.

For example... dat <- read.csv("C:/Users/yourname/yourproject/data/mydata.csv") isn't going to work on my computer.

Code Style

Be consistent with your R code style. Here are a couple links to Google and Hadley Wickam's R code style guidelines.

https://google.github.io/styleguide/Rguide.xml

http://adv-r.had.co.nz/Style.html

Comment, Comment, Comment

It never hurts to add comments.
Generally I comment on why I am doing something.

Use "echo=FALSE" in your RMarkdown chunks

Generally speaking, you don't want to show your R code in your report. If you were writing an article about your new R package, well then show the code. If you add "echo=FALSE" to your code chunk, your commands won't show up in your output.

Project Chunk Options.

There are many code chunk options. http://yihui.name/knitr/options/

You can set chunk option defualts using the opts_chunk() function. This is useful so you don't have to repeat yourself along the way with things like standard figure sizes and turning warnings and messages off. If you do this, make it the first chunk of your markdown file. Note that you will need to reference the knitr library.

```
```{r echo=FALSE}

library(knitr)

opts_chunk$set(fig.width=4, fig.height=4, fig.path='figure/',

         echo=FALSE, warning=FALSE, message=FALSE)

```
```

This particular set will load the knitr library and set chunk option defaults for figure size, figure location as well as turn echoes, warnings, and messages off.

One or Many Scripts

Decide on whether you will be using one single script or multiple scripts for your project.

A single r markdown file can be used to hold all of your R code and keep everything in one nice tidy package. All you need to do is click the "Knit " button in R Studio and everthing will run.

However, I generally use multiple scripts to keep things logically separated. I might have a script for any special functions, a script for data import, one for data cleaning, one for analysis, one for figure generation, and finally a markdown file to create the report. Sometimes I might combine the import and cleaning steps into a single script and the analysis and figures into another.

If you use multiple scripts, can't just source() them in to your markdown file and expect them work. For some reason the sourced in R files and the R Markdown file create different environments (i.e. locations that the variables get stored)

There are two work arounds to this.

1.) Run your scripts first and from within your R scripts, save all of the important objects (variables, tables, figures)

```
# from within your R file... "project.R"
save(x,y,z,fig1,table1, file="project.Rdata")
```

then from in your R markdown file, source the R file and then load the saved objects and do something with them.

```
source("project.R")
load("project.Rdata")
fig1
```

This has the added benefit of keeping all of those annoying package loading messages from appearing in your report.

2.) The second option is to render/knit your final report from a command in your R script rather than using the Knit button in R Studio. Just be sure to include the rmarkdown library in your R code.

```
# from within your R file...

# A bunch of R code.
a <- 1:20
plot(a)
library(rmarkdown)
render("myReport.Rmd", "html_document")
render("myReport.Rmd", "pdf_document")
render("myReport.Rmd", "all")
```

If you have multiple R files and a final Rmd file. It might be worth creating a single R file that can be used as the master R file to source everything in the proper order.

```
library(rmarkdown)
source("import.R")
source("clean.R")
source("analysis.R")
source("plots.R")
render("report.Rmd", "pdf_document")
```

Then to create your report you simply source this master R file.

Required Packages

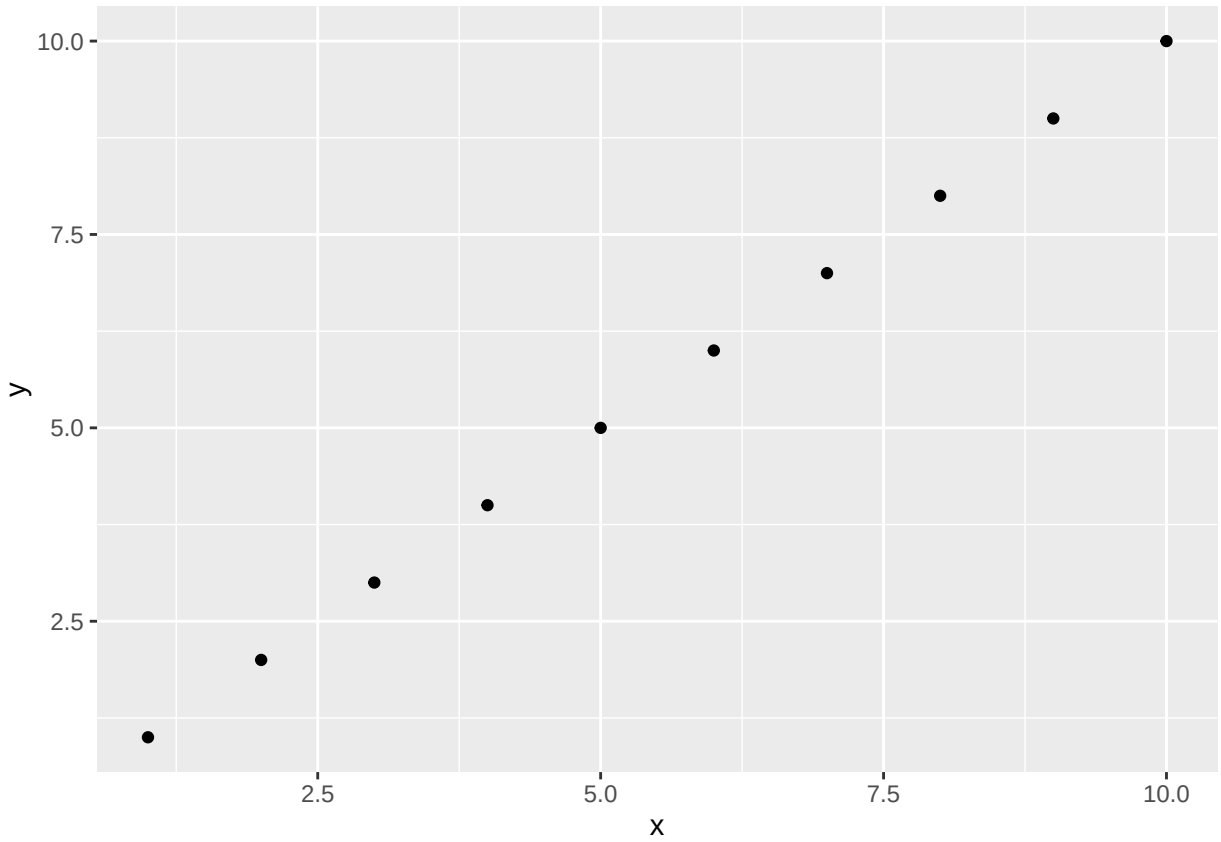
Chances are not everyone is going to have the same packages installed. If you know which packages are needed, you can include the following bit of code to test for installed packages, install missing ones, and then load them.

Using the chunk option of "include=FALSE" will allow the code to be run (i.e. install and load the libraries) without any of the annoying package messages that don't seem to understand "message=FALSE" and would otherwise show up in your report output.

Saving Plots as Objects

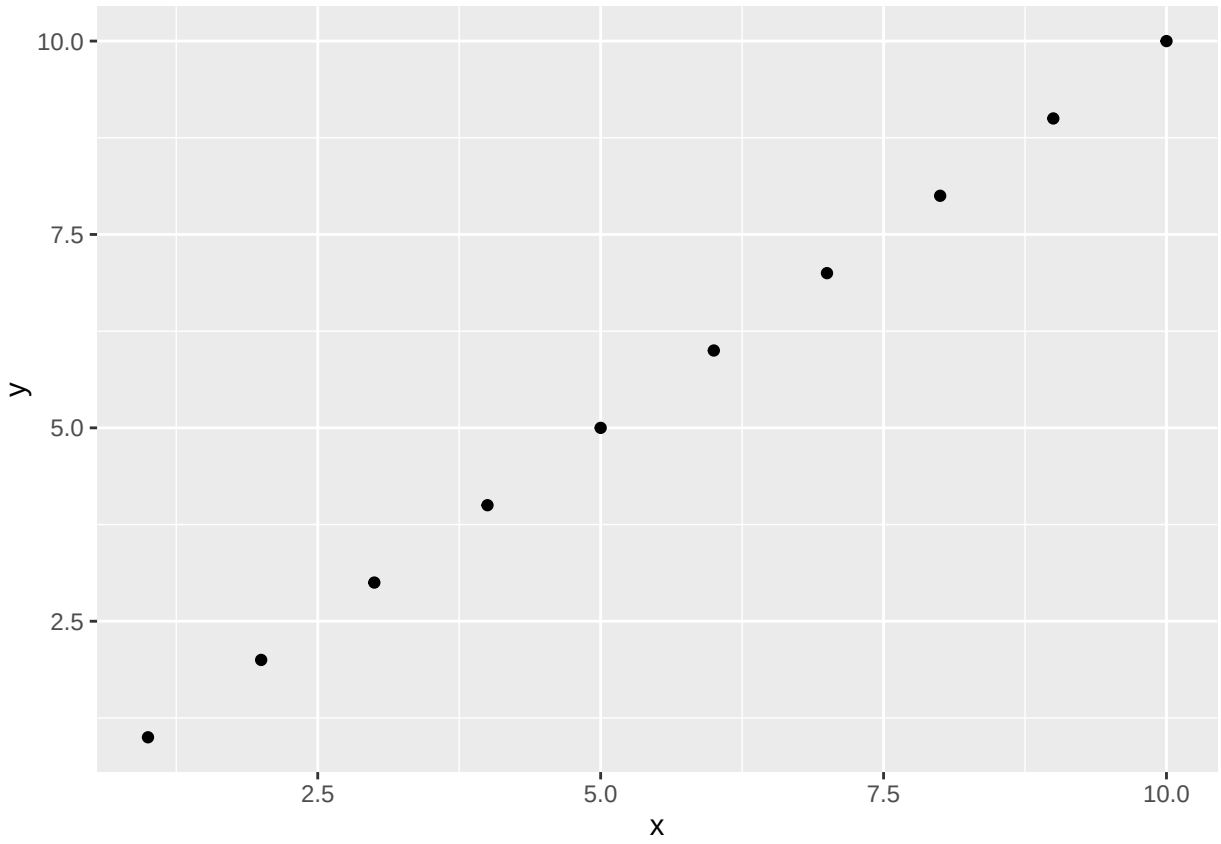
When you use ggplot2, you may have noticed that it is common to assign a ggplot to an object and then call that object to get the ggplot to show. If you use informative object names than each figure in your report can be called at will.

```
library(ggplot2)
mydata <- data.frame(x=1:10, y=1:10)
fig1<-ggplot(dat=mydata, aes(x,y)) + geom_point()
fig1
```



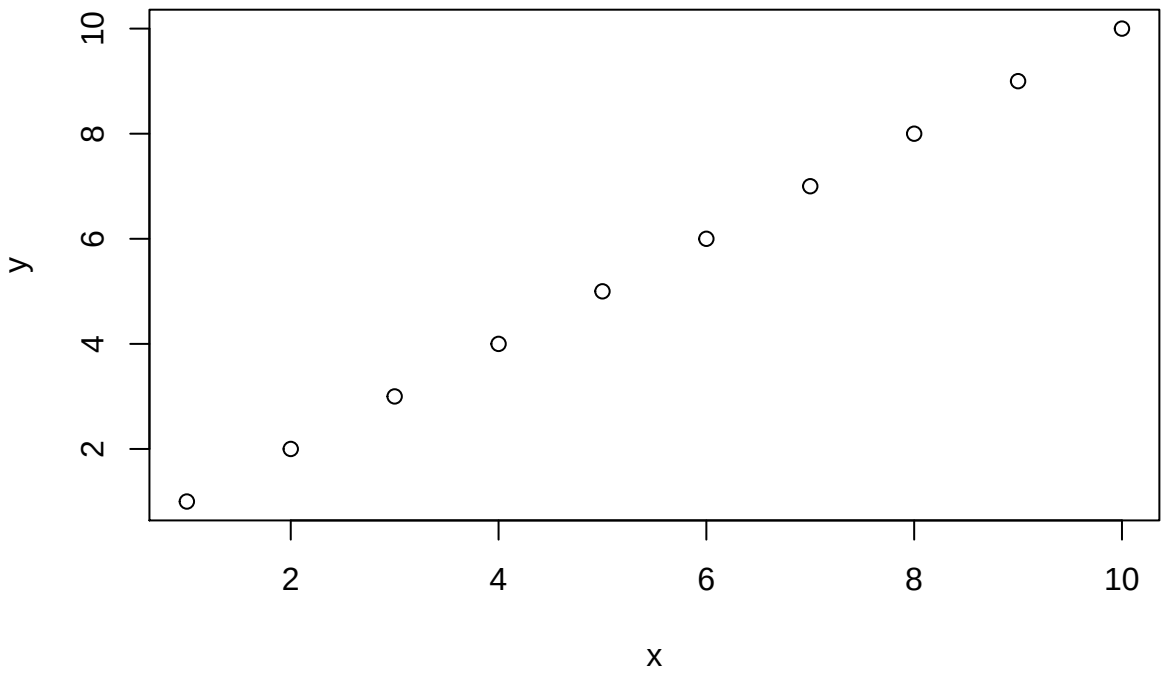
This is useful if you have all of your analysis and figure generation in an R script. You can save the figure objects and load them in to your R Markdown file where ever you want them.

```
fig1
```

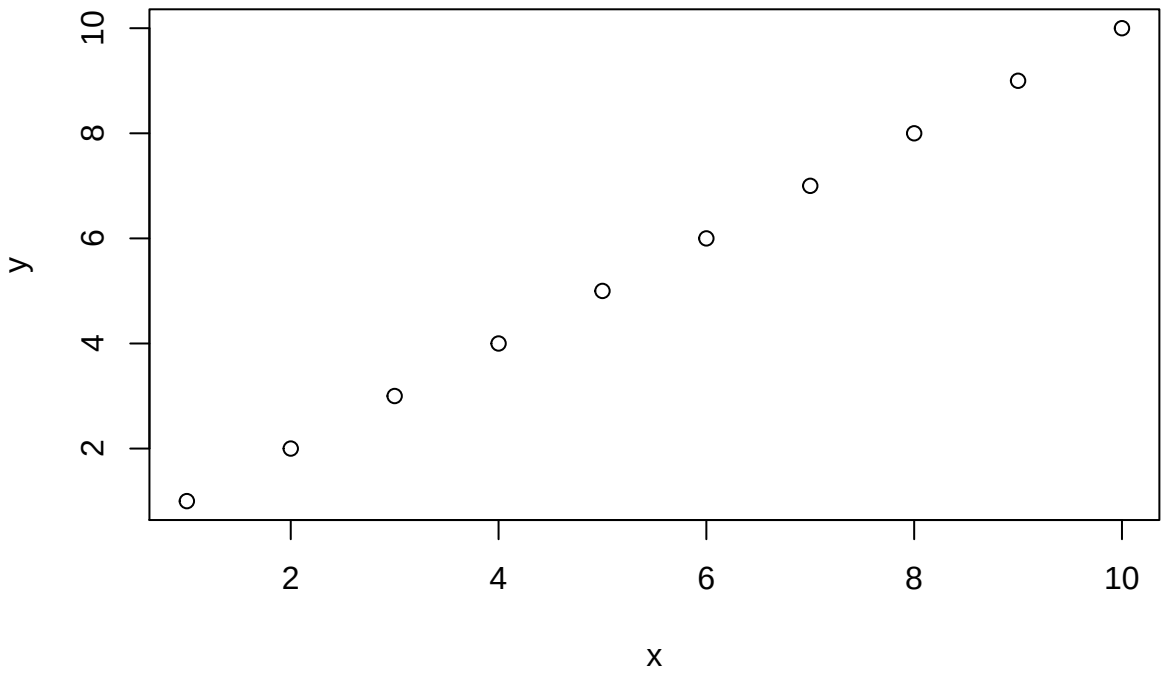


Most standard plots (i.e. not ggplot2 plots) don't save to objects directly. If you want to do this with a regular plot you can record your current plot to an object using recordPlot(). Once recorded, the object can be called back at any time.

```
plot(mydata)
```



```
fig2 <- recordPlot()
plot.new() ## clean up device
fig2 # redraw
```



Formatting Tables

If you just print your standard R data.frames in your report they will look ugly.

```
n <- 100
x <- rnorm(n)
y <- 2*x + rnorm(n)
out <- lm(y ~ x)
summary(out)$coef
```

```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.117815 0.09074963  1.298242 1.972511e-01
## x           1.929862 0.08736661 22.089241 7.754707e-40
```

You can use the kable() function to wrap tables for prettier output in html or pdf.

```
n <- 100
x <- rnorm(n)
y <- 2*x + rnorm(n)
out <- lm(y ~ x)
library(knitr)
kable(summary(out)$coef, digits=2)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.02	0.1	0.17	0.87
x	2.26	0.1	21.86	0.00

Kable will create HTML or LaTeX tables depending on how you knit your markdown file within R Studio. If you are kniting to HTML output you can add CSS to your R markdown file to further control the output format.

Embedded LaTeX

Sometimes you might need to add a mathematical equation to your report.

```
$$ \sqrt{27} $$
$$ \alpha $$
$$ \displaystyle \int_0^{\infty} f(x) \; dx $$
$$ \frac{d}{dx}\sin x=\cos x $$
$$ \frac{n!}{r!(n-r)!} $$
$$ 2H_2 + O_2 \xrightarrow{n,m}2H_2O
```

$$\sqrt{27}$$

$$\alpha$$

$$\int_0^{\infty} f(x) \; dx$$

$$\frac{d}{dx}\sin x=\cos x$$

$$\frac{n!}{r!(n-r)!}$$

$$2H_2 + O_2 \xrightarrow{n,m} 2H_2O$$

LaTeX equation formatting is tricky. There are many references out there to help and I found at least one online equation editor. But even that isn't easy to use.

https://www.codecogs.com/latex/eqneditor.php

Report Output Format (HTML, PDF, DOC)

Each of the knitr output options has it's advantages.

HTML is great if you're putting something on the web. I use this for putting stuff on to D2L. If you know a bit about style sheets (CSS) you can have great control on how the output looks.

DOC is good if you need to create a report that someone else will edit.

PDF gives you most control on output based on LaTeX, but there is a steep learning curve. If you're lucky someone will have created a LaTeX style you can incorporate into your report. Several publishers provide these styles for some of their journals.