

Introduction to R: A small program

Steve Simon

Created 2020-01-27

Special note

- This PowerPoint slide show was created using R.
 - Not complicated
 - But beyond scope of this class
- Source
 - <https://github.com/pmean/introduction-to-r/tree/master/part1/src>
- A second resource
 - <http://blog.pmean.com/powerpoint-with-r-markdown/>

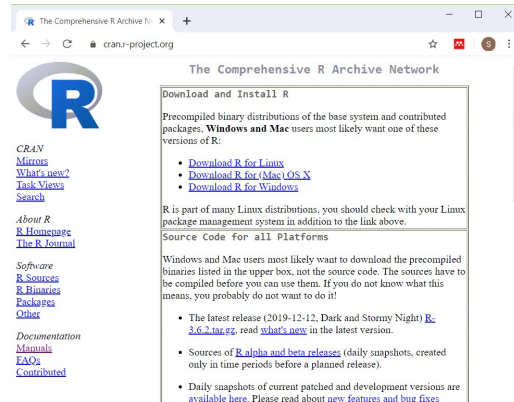
If you are viewing this PowerPoint presentation, I need to point out that it was developed using R. The process is not very complicated, but it is beyond the scope of this class.

If you are curious, you can look at the code that I used to develop this PowerPoint presentation. Or you can watch a short video on how this works.

But don't feel obligated to look at it. You will not be responsible for any of this in an introductory class.

It may seem a bit weird to have an R program that creates a PowerPoint presentation that talks about a different R program, but it works well for me.

Installing R (<https://cran.r-project.org/>)

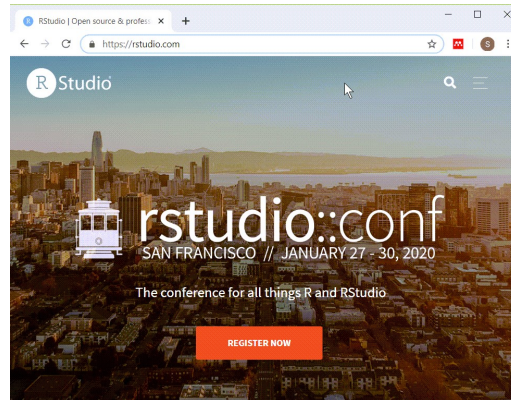


Screenshot of webpage for installation of R

Here is the main page for CRAN. CRAN stands for the Comprehensive R Archive Network. This is where you can download a Linux, Mac, or Windows version of R. Ignore the source code. That is only needed for very advanced applications.

Installing RStudio

(<https://rstudio.com/>)



Screenshot of main page for RStudio

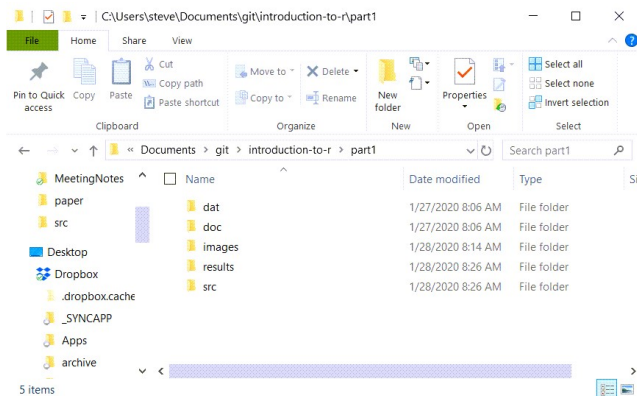
Installing R and R Studio

- R is required
- RStudio is strongly recommended
- Do not delay in getting this software installed
- Find me if you have ANY problems

It should be very easy to install R and RStudio on your computer, but don't wait. Sometimes installations can get hung up and you won't be able to make any progress in this class without first getting the software installed.

If you have any problems at all with installation, see me right away.

Recommended directory structure



Screenshot of the directory structure I use

Here is the directory structure that I use. You do not have to follow this structure, but it is recommended, not by me, but by the expert programmers at a group known as Software Carpentry. I'll elaborate in greater detail about this later, but wanted to mention it now. If you are relatively new to programming, you want to start off using good programming practices. A standardized directory structure helps a lot with this especially if you are working with others.

“A place for everything, everything in its place”

- dat
 - raw/intermediate data files
- doc
 - documentation
- images
 - graphs
- results
 - program output
- src
 - program code

The quote at the top of the slide is an organizational principle espoused by Benjamin Franklin. If you’ve seen my office, you’ll know that I am probably the last person to lecture you on organization. But I have found that a standardized directory structure has made my life a lot easier.

The dat folder contains any raw data files. It’s also where I put intermediate files, files that I create and save for later re-use. Some people put intermediate files in the results folder, and that’s a fine alternative. Just be sure to be consistent about it.

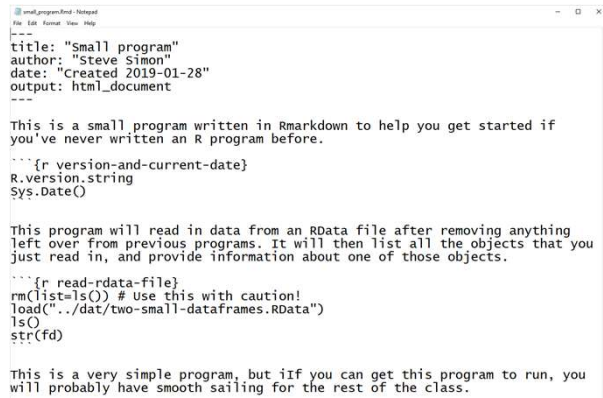
The doc folder contains any documentation associated with the work. The IRB approved protocol, if I have one, will go here. If I am working with someone and they send me a paper that helps describe the type of data analysis they want, I put it in this folder. I also print key emails from the other team members to pdf format and store them here as well.

If my programs produce any graphs, I will store them in the images folder. I use screenshots of various web pages a lot in my work and I put those here as well.

I usually store program output in the results folder, but not always, for reasons I don’t want to get into.

The program code goes in the src folder.

Anatomy of a small R program, overview

A screenshot of a text editor window titled 'small_program.Rmd - Notepad'. The window contains Rmarkdown code. It starts with a YAML header: 'title: "Small program"', 'author: "Steve Simon"', 'date: "Created 2019-01-28"', and 'output: html_document'. This is followed by a paragraph: 'This is a small program written in Rmarkdown to help you get started if you've never written an R program before.' Then there is an R code chunk: '```{r version-and-current-date}' followed by 'R.version.string' and 'Sys.Date()'. Another paragraph follows: 'This program will read in data from an RData file after removing anything left over from previous programs. It will then list all the objects that you just read in, and provide information about one of those objects.' This is followed by another R code chunk: '```{r read-rdata-file}' followed by 'rm(list=ls()) # Use this with caution!', 'load("../dat/two-small-dataframes.RData")', 'ls()', and 'str(fd)'. The final paragraph says: 'This is a very simple program, but if you can get this program to run, you will probably have smooth sailing for the rest of the class.'

Full listing of small_program.Rmd

This is a full listing of a small program written in Rmarkdown. The font is too small to read. Don't worry, I just wanted you to see the full picture. I'll look at small pieces of this program using a readable font size.

Anatomy of a small R program, header

– Rmarkdown code

```
---  
title: "Small program"  
author: "Steve Simon"  
date: "Created 2019-01-28"  
output:  
  html_document: default  
---
```

Here is the typical header for an Rmarkdown program.

Anatomy of a small program, free text comments

– Rmarkdown code

```
This is a small program written in Rmarkdown to  
help you get started if you've never written an R  
program before.
```

Anatomy of a small program, code chunk

– Rmarkdown code

```
```{r version-and-current-date}  
R.version.string
Sys.Date()
```
```

Anatomy of a small program, second set of free text comments

– Rmarkdown code

```
This program will read in data from an RData file  
after removing anything left over from previous  
programs. It will then list all the objects that  
you just read in, and provide information about  
one of those objects.
```

Anatomy of a small program, second code chunk

– Rmarkdown code

```
```{r read-rdata-file}
rm(list=ls()) # Use this with caution!
load("../data/two-small-dataframes.RData")
ls()
str(bump)
```
```

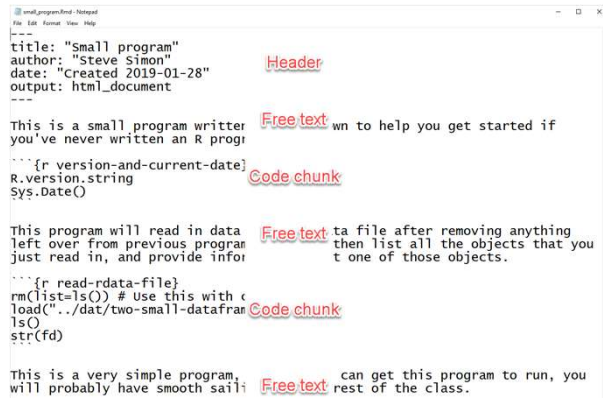
Anatomy of a small program, third set of free text comments

– Rmarkdown code

```
This is a very simple program, but if you can get  
this program to run, you will probably have  
smooth sailing for the rest of the class.
```

After the second chunk of R code, I add a couple of extra comments.

Anatomy of a small program, review



The screenshot shows a text editor window titled 'small_program.Rmd - RStudio'. The content is an R Markdown document with the following structure:

```
---
title: "Small program"
author: "Steve Simon"
date: "Created 2019-01-28"
output: html_document
---

This is a small program written to help you get started if
you've never written an R program.

```{r version-and-current-date}
R.version.string
Sys.Date()
```

This program will read in data left over from previous programs,
just read in, and provide information. It will read in a file after removing anything
then list all the objects that you
t one of those objects.

```{r read-rdata-file}
rm(list=ls()) # Use this with care
load("../dat/two-small-dataframes.Rdata")
ls()
str(fdata)
```

This is a very simple program, can get this program to run, you
will probably have smooth sailing. rest of the class.
```

Annotations in red text are placed to the right of the document content:

- Header**: points to the YAML header.
- Free text**: points to the introductory paragraph.
- Code chunk**: points to the first R code block.
- Free text**: points to the paragraph describing the program's purpose.
- Code chunk**: points to the second R code block.
- Free text**: points to the final paragraph.

Full listing of small_program.Rmd with annotations

Here is the full program again. You can see that it starts with a header, and alternates between free text and R code.

Program output, overview



```
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
## [1] "1000-01-01"
```

Screenshot of entire output

Here is the full output. The fonts are too small to read, so let's zoom in again on individual pieces.

Program output, header

Small program

Steve Simon

Created 2019-01-28

Screenshot of output header

This is the header. It displays the title, author, and date from the code header.

Program output, free text

This is a small program written in Rmarkdown to help you get started if you've never written an R program before.

Screenshot of output free text

This is first set of free text. Rmarkdown will reflow the text, if needed, to fit inside the margins.

Program output, code chunk

```
R.version.string
```

```
## [1] "R version 3.6.1 (2019-07-05)"
```

```
Sys.Date()
```

```
## [1] "2020-01-28"
```

Screenshot of output code chunk

This is the first code chunk. By default, Rmarkdown will display the individual commands in a gray box, followed by the output in a white box.

Program output, more free text

This program will read in data from an RData file after removing anything left over from previous programs. It will then list all the objects that you just read in, and provide information about one of those objects.

Screenshot of output free text

This is second set of free text.

Program output, second code chunk

```
rm(list=ls()) # Use this with caution!
load("../dat/two-small-dataframes.RData")
ls()
```

```
## [1] "fd" "fo"
```

```
str(fd)
```

```
## 'data.frame':    252 obs. of  19 variables:
##  $ case   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ fat.b   : num  12.6 6.9 24.6 10.9 27.8 20.6 19 12.8 5.1 12 ...
##  $ fat.s   : num  13.3 6.1 23.3 10.4 28.7 20.9 19.2 12.4 4.1 11.7 ...
##  $ dens   : num  1.07 1.09 1.04 1.08 1.09 ...
##  $ age     : int  23 22 22 26 24 24 26 25 25 23 ...
##  $ wt      : num  154 173 154 135 154 ...
##  $ ht      : num  67.8 72.2 66.2 72.2 71.2 ...
##  $ bmi     : num  23.7 23.4 24.7 24.9 25.6 26.5 26.2 23.6 24.6 25.8 ...
##  $ cfw     : num  135 161 116 169 133 ...
##  $ neck    : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 42.1 ...
##  $ chest   : num  93.1 93.6 95.8 101.8 97.3 ...
##  $ abdomn  : num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5 88.6 ...
##  $ hip     : num  94.5 98.7 89.2 101.2 101.9 ...
##  $ thigh   : num  89 88.7 89.6 60.1 63.2 66 88.4 60 62.9 63.1 ...
##  $ knee    : num  37.3 37.3 38.9 37.3 42.2 42 38.3 39.4 38.3 41.7 ...
##  $ ankle   : num  21.9 23.4 24 22.8 24 25.6 22.9 23.2 23.8 25 ...
##  $ biceps  : num  32 30.8 28.8 32.4 32.2 35.7 31.9 30.5 35.9 38.6 ...
##  $ forearm : num  27.4 28.9 25.2 29.4 27.7 30.6 27.8 29 31.1 30 ...
##  $ wrist   : num  17.1 18.2 16.6 18.2 17.7 19.8 17.7 18.8 18.2 19.2 ...
```

Screenshot of output code chunk

This is the second code chunk. It displays the title, author, and date from the code header.

Program output, last set of free text

This is a very simple program, but if you can get this program to run, you will probably have smooth sailing for the rest of the class.

Screenshot of output free text

This is last set of free text.

Review

- In this video, you saw
 - Installation pages for R, RStudio
 - Anatomy of a small program
 - Output from the small program
- Review Canvas for the work assigned to this module

This is a very basic start. You saw the web pages where you install R and RStudio from. Then you saw the pieces of a small program written in Rmarkdown, as well as the output.

Take a look at Canvas, please, to see the work that you need to do for this module.