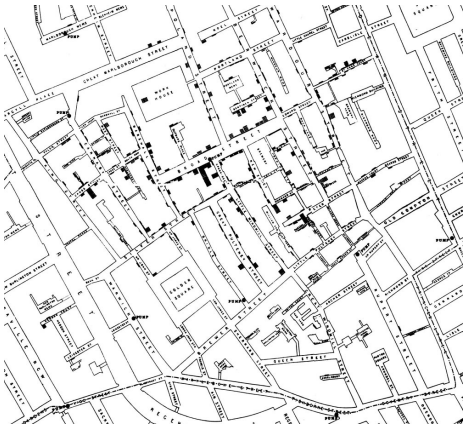
EPI 563: Spatial Epidemiology, Fall 2020

Michael Kramer

Last updated: 2020-08-08

Contents

How to use this eBook



Welcome to *Concepts & Applications in Spatial Epidemiology (EPI 563)*! This eBook is one of several sources of information and support for your progress through the semester. For an overview of the course, expectations, learning objectives, assignments, and grading, please review the full course syllabus on Canvas. This eBook serves to provide a *'jumping off point'* for the content to be covered each week. Specifically, the content herein will introduce key themes, new vocabulary, and provide some additional detail that is complementary to the *asynchronous* (pre-recorded) video lectures, and foundational to the *synchronous* (in class) work. As such you should plan to read the assigned content **before** each weekly session.

You can read the content on the web via your browser, or optionally, you can download a PDF of the *current* content. However please note that I will be continually updating the eBook throughout the semester, so if you choose to download, please double-check the **Last updated** date to be sure you have the most recent version.

Part I

Getting ready...

Software installation

The information in this module follow on the pre-class video on setting up `R` and `RStudio` on your computer.

Installing `R` on your computer

As of August 2020, the most up-to-date version of `R` is 4.0. The *R Project for Statistical Computing* are continually working to update and improve `R`, and as a result there are new versions 1-2 times per year.

If you already have `R` installed, you can open the console and check your current version by doing this: `R.Version()$version.string`

If you do not have `R` or have an older version than 3.5.1 you can install `R` by going to the `R` repository: https://www.r-project.org/. Note that there are many 'mirrors' or servers where the software is stored. Generally it is wise to select one that is geographically close to you, although any should work in theory. One mirror that is relatively close to Atlanta is here: http://archive.linux.duke.edu/cran/

Installing RStudio on your computer

The current version of RStudio 1.3.1056. If you do not have RStudio or have a version older than 1.2 please install/update.

TO INSTALL: go to https://www.rstudio.com/products/rstudio/download/

TO UPDATE: Open RStudio and go to Help Menu and choose 'Check for Updates'

Installing packages for this course

While base `R` has a great deal of essential functionality, most of the power of `R` comes from the rapidly growing list of user-created and contributed 'packages'. A package is simply a bundle of functions and tools, sometimes also including example datasets, basic documentation, and even tutorial 'vignettes'. You can see all the official `R` packages by going here: https://cran.r-project.org/web/packages/.

The most common way to install package in `R` is with the `install.pacakges()` command. For instance to install the package `ggplot2` you do this:

```
install.packages("ggplot2")
```

Notice that for `install.packages()` you need quotes around the package name. Remember that you only need to install a package once (although you may have to update packages occasionally – see the green Update button in the Packages tab in R Studio). When you want to actually use a package (for example `ggplot2`) you call it like this:

```
library(ggplot2)
```

Notice that for the `library()` function you **do not** need quotes around the package name (unlike the `install.packages()` above). If your call to `library()` is working, nothing visible happens. However if you see errors, they might be because your package is out of date (and thus needs to be updated/reinstalled), or because some important *dependencies* are missing. Dependencies are other packages on which this package depends. Typically these are installed by default, but sometimes something is missing. If so, simply install the missing package and then try calling `library(ggplot2)` again.

While **most** packages can be installed as mentioned above (e.g. using `install.packages()`), there are instances where an installation requires additional *tools*, for instance to install from source or from github. Luckily there is a package for that! It is called `Rtools`, and you should install that **before** you install the packages below.

As you submit each installation request, note the output. If you get a warning that says installation was not possible because you are missing a package *'namespace'*, that suggests you are missing a dependency. Try installing the pacakge mentioned in the error. If you have trouble, reach out to the TA's!

Installing `Rtools40`

Navigate to this website: https://cran.r-project.org/bin/windows/Rtools/ and follow the instructions specific to your operating system.

Installing packages used for general data science

For the rest of this page, copy and paste the provided code in order to install packages necessary for this course.

These packages will support some of our general work in `R`, including working with `RMarkdown` and R Notebooks, as well as data manipulation tools from the `tidyverse`. You can learn more about the `tidyverse` here: https://tidyverse.tidyverse.org/. The `tidyverse` is actually a collection of data science tools including the visualization/plotting pakcage `ggplot2` and the data manipulation package `dplyr`. The package `tinytex`, `rmarkdown`, and `knitr` are all necessary for creating `R` Notebooks, which is the format by which many assignments will be submitted.

```
install.packages('tidyverse')
install.packages(c('tinytex', 'rmarkdown', 'knitr'))
tinytex::install_tinytex()
# this function installs the tinytex LaTex on your
#  computer which is necessary for rendering (creating) PDF's
```

Installing packages use for geographic data

There are many ways to get the data we want for spatial epidemiology into `R`. Because we often (but don't always) use census geographies as aggregating units, and census populations as denominators, the following packages will be useful. They are designed to quickly extract both geographic boundary files as well as attribute data from the US Census website via an API. **NOTE**: For these to work you have to request a free Census API key. Notice the `help()` function below to get instructions on how to do this.

```
install.packages(c('tidycensus','tigris'))

help('census_api_key','tidycensus')
```

Installing packages used for spatial data manipulation & visualization

This section installs a set of tools specific to our goals of importing, exporting, manipulating, visualizing, and analyzing spatial data. The first line of packages have functions for defining, importing, exporting, and manipulating spatial data. The second line has some tools we will use for visualizing spatial data (e.g. making maps!).

```
install.packages(c('sp', 'sf', 'rgdal', 'rgeos', 'maptools', 'OpenStreetMap'))
install.packages(c('tmap', 'tmaptools', 'ggmap', 'shinyjs', 'shiny', 'micromap'))
```

Installing packages used for spatial analysis

Finally these are packages specifically to spatial analysis tasks we'll carry out.

```
install.packages(c('spdep', 'CARBayes', 'sparr'))
install.packages(c('GWmodel', 'spgwr') )
```

Part II

Weekly Modules

Chapter 1

Locating Spatial Epidemiology

Part III

Appendices

Tips for `R` Notebooks

Additional Resources

- R Markdown Cheatsheet
- Comprehensive guide to using R Markdown
  - Chapter within the R Markdown guide specific to Notebooks

Why `R Notebooks`?

For most assignments in this course, at least a portion of the deliverable will be a fully-functional, annotated `R Notebook`. These *notebooks* are actually a specific case of `rmarkdown` which itself is a format for creating reproducible documents with interspersed `R` code, analytic results and text. For example this eBook, and many other resources in this course are created using `rmarkdown` or related packages such as `bookdown`.

But as I said, `R Notebooks` are a specific instance or case of markdown that is incorporated into R Studio and has some nice features for the applied data analyst.

- Notebooks contain text that explains what is happening, interprets findings, or notes areas in need of further exploration
- Notebooks contain functional `R` code that when run places the results **inside** the Notebook document
- Notebooks work in an *interactive* mode. This means that as you are coding and working you can see the results in the document. When you save the Notebook the text, the code **and the results** are saved!

So the reason for using Notebooks is that they provide a means for clear annotation and documentation combined with ready reproducitiblity. Reproducibility means that someone else (or a future you!) could come back and get the same result again.

To benefit from the advantages above, I recommend you gain familiarity with the basic (and perhaps the optional) formatting described below. I also recommend you develop a knack for rich annotation and documentation, not just brief (often cryptic) comments that we are all used to writing in SAS and other code! Document what you **plan to do**. Document what you **did**. Document what the **results means**. Document what else **needs to be done**.

What you need to know

This file summarizes both **important** and just a small handful of **optional** functions for effectively using R Notebooks. The **important** functions are those necessary to effectively intersperse narrative text and description communicating what you did and what it means, with clear R code and the resulting output. The **optional** parts are about some simple formatting tools. These **are not** necessary for your homework (our goal is documentation of analytic process not being 'pretty'), but you may find them useful.

Important R Notebook functions

The YAML

```
---
title: "Title of your notebook"
author: "Your Name Here"
date: "Submission date here"
output:
  html_notebook:
    number_sections: yes
    toc: yes
    toc_float: yes
---
```

When you create a new R Notebook or R Markdown file from within R Studio (e.g. via *File > New File > R Notebook*), a 'YAML' will automatically be created at the top of the script delineated by three dash lines `---`. *YAML* stands for "*yet another markup language*" and it is a set of instructions about how the finished notebook will look and be structured. You can accept the default YAML structure (of course modifying the title) or copy/paste the YAML from the top of this script. You can also read more online about additional customizations to the YAML, but none are necessary for this course.

Typing text

The utility of R Notebooks is the ability to more completely document your thinking and your process as you carry out analyses. It is not necessary to be wordy just for the sake of taking up space, but this is an opportunity to clearly delineate goals, steps, data sources, interpretations, etc.

You can just start typing text in the script to serve this purpose. Some text formatting functions are summarized later in this document, and in Cheatsheets and online resources linked to elsehwhere.

Adding R Code

R Notebooks let you write R code within your Markdown file, and then run that code, seeing the results appear right under the code (rather than only in the Console, where they usually appear).

There are 2 ways to add a new chunk of R code:

1. Click the green `C-Insert` button at the top of the editor panel in R Studio. The top option is for R code.

2. Use a keyboard short cut:

    - *Mac* Command + Shift + I
    - *Windows* Ctrl + Alt + I

Notice these R code chunks are delineated by three back-ticks (sort of like apostrophers)...these back-ticks are typically on the same key as the tilde (~) on the upper left of most keyboards. The space between the sets of 3 back-ticks is where the R code goes and this is called a *code chunk*.

Inside these 'chunks' you can type R code just as you would in a regular R script.

When you want to run it you can either:

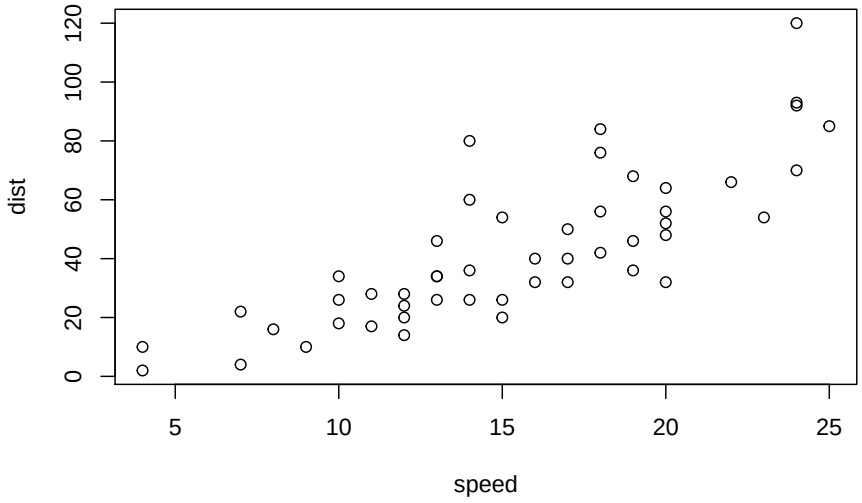
1. Place your cursor on a line and click Ctrl+enter (Windows) or CMD+Return (Mac), or you can click the *Run* button at th etop of the editor pane in R Studio.
2. To run all of the code within a chunk click the green *Run Current Chunk* button at the upper-right of the code chunk.

Below is some code and the corresponding results.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
plot(cars)
```



In this way you can iterate through your analytic process…switching between running code, viewing output, documenting in free text.

If you want to see what your current work-in-progress looks like in HTML, you can click the *Preview* button at the top of the panel. This will both save your document, and open the *Viewer* panel.

Making tables

While not required, you may want to summarize data in a table in `R` Markdown. There are packages devoted to creating tables, but you can create a quick-and-dirty table just using keyboard symbols.

1. First start by making a header row. Separate each column name with a *'pipe'* symbol, `|`
2. Put a continuous line of dashes (`-----`) under each column name, separating columns with pipe symbole (`|`)
3. Now type text corresponding to each row and column. Separate columns with pipe (`|`) and separate rows by carriage return/Enter

So the following text typed directly into the Markdown file (e.g. not inside a code chunk):

```
Column 1  | Column 2 | Column 3
----------|----------|-----------
Text 1    | Text 2   | Text 3
Next line | Next line 2 | Next line 3
```

Will produce the following output:

Column 1	Column 2	Column 3
Text 1	Text 2	Text 3
Next line	Next line 2	Next line 3

Workflow

The benefit of Notebooks (slightly different from regular Markdown) is that you can work interactively with your code, seeing results immediately just as you would with a regular script. In contrast a 'regular' Markdown file doesn't run the code until you click 'Knit'.

Here is what I recommend for workflow:

1. Name your script and save it to the desired location. Wherever you save it will become the default Working Directory for any code run from *within the Notebook*.

2. Carry out your analysis, inserting code chunks, running them, and documenting them as you go.

3. As a final check of *reproducibility* (the assurance that your code is self-contained and not dependent on steps you did outside the script) I recommend you always end by clicking the *RUN* button at the top of the panel. Specifically, choose **Restart R and Run all Chunks**. If there is an error when you did this, then something is missing in your code.

Optional functions

The list of formatting functions is long. I include only a couple I find useful (but not mandatory) here:

Customizing your YAML

While the default YAML is perfectly fine, the YAML at the top of this script includes a few added functions including:

1. Specify a table of contents - this only works if you use headers
2. Specify section numbering
3. Specify that the table of contents should be 'floating' which means that in *html* it is visible even when you scroll. For PDF rendering, 'float' is not an option.

Simple formatting of your Notebook

Often it is helpful use headers to separate tasks or steps in your code. You can easily create headers using the hastag/pound sign `#`. Specifically…

- `#` at the beginning of the line denotes a top-level (level-1) header that will be large and bold.
- `##` at the beginning of the line denotes level-2 header
- `###` unsurprisingly is a level-3 header!
- Make sure there is a space between the `#` and the text
- Always leave a blank line (return/enter) between the header text and the '*regular*' text.

You can also make numbered or bulleted lists if that is helpful. A line that begins with either an asterisk (`*`) or a number will begin a bulleted or numbered list.

Headers are populated into the table of contents, if specified.

Text formatting

The `R` Markdown Cheatsheets have lots of examples of formatting. Three things that I use more frequently are bold, italics, and numbered or bulleted lists.

Key stroke	Result
`*italics*`	*italics*
`**bold**`	**bold**

1. Numbered lists start with number, and each line must end with 2 space (or have blank line between).
2. Instead of numbers you can use letters

- Bulleted lists can be initiated with an asterisk or +, and also must have 2 spaces (or blank carriage return) at end of each item.

Final Note

Remember that a final step when you think you are done with a project, is to Click `Restart R and Run all Chunks`, and then save/preview the Notebook **after** doing this to be sure it is what you expect.