Example Report

Thomas Brambor

2018-09-17

Contents

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

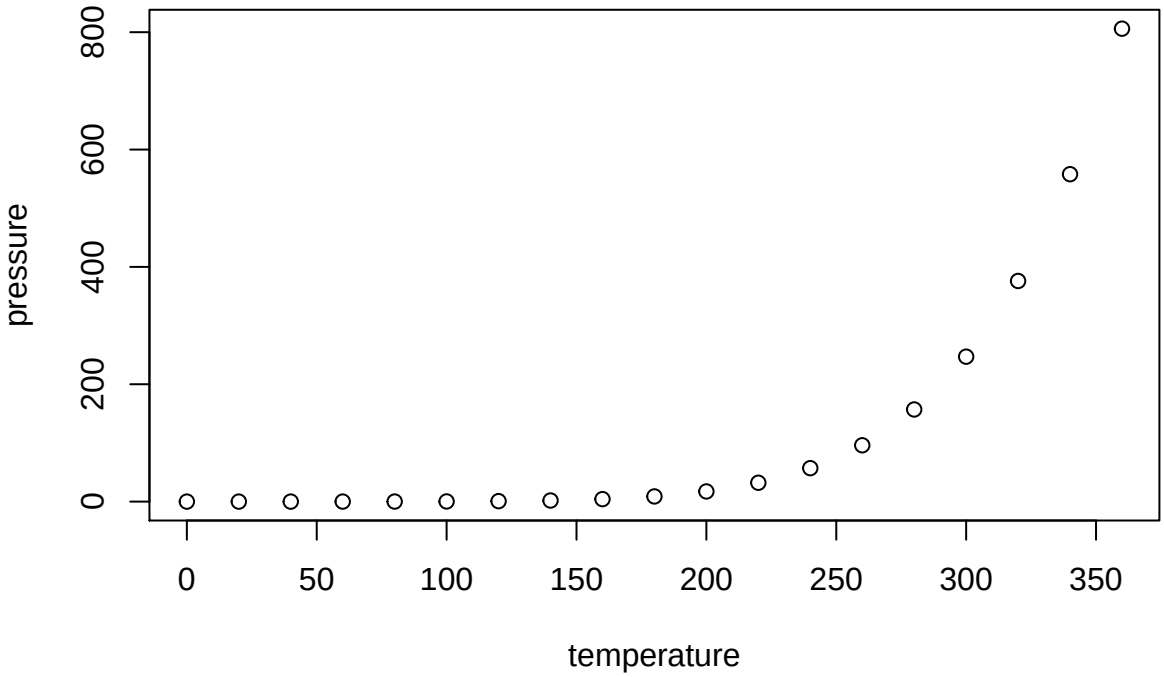
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Add other types of code with `engine`

By default, your R notebook chunks will be run using R. However, it's entirely possible to write chunks that use other engines to execute. For instance, you can add some Python to your notebook:

```
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print "iteration {iteration} is {name}".format(
    iteration=i, name=name)
```

```
## iteration 0 is john
## iteration 1 is pat
## iteration 2 is gary
## iteration 3 is michael
```

In-line Latex and R code

You can convert the temperature unit from Kelvin to Celsius with the formula

$$celsius = kelvin - 273.15$$

And you can convert the result to Fahrenheit with the formula

$$fahrenheit = celsius \times \frac{9}{5} + 32$$

For example, degrees Kelvin corresponds to 9 degrees Celsius.

Interactive Visualizations

We can use interactive features as well. Let's load the required packages first.

HTML Widgets

If your R analysis involves interactive components, you're probably already familiar with the `htmlwidgets` library. These, too, are supported in the notebook. Run this chunk to see an interactive graph:

```
dygraph(nhtemp, main = "New Haven Temperatures") %>%
  dyRangeSelector(dateWindow = c("1920-01-01", "1960-01-01"))
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please
```

Running Code

One of the goals of the notebook is to provide a seamless environment for interacting with R – that is, you shouldn't need to reach for the console, even though chunks send code there. To help you see the progress of your chunk – that is, which lines have been executed and which haven't – RStudio draws an indicator in the editor gutter. Try running this chunk:

```
Sys.sleep(1); runif(3)
```

```
## [1] 0.7338701 0.9208472 0.4459780
```

```
Sys.sleep(1); runif(3)
```

```
## [1] 0.4020128 0.8544962 0.1477301
```

```
Sys.sleep(1); runif(3)
```

```
## [1] 0.0542959 0.2780038 0.5515706
```

```
Sys.sleep(1); runif(3)
```

```
## [1] 0.2668870 0.5414724 0.5827051
```

```
Sys.sleep(1); runif(3)
```

```
## [1] 0.2254768 0.1797464 0.9802712
```

Stepwise Execution

Sometimes you may want to run portions of your chunk rather than the whole thing. That's just fine too. Try using *Ctrl+Enter* (OS X: *Cmd+Enter*) to run this chunk line by line.

```
cities <- read.csv("cities.csv")
cities
```

```
##   X          City     Lat     Long     Pop
## 1 1        Boston 42.3601 -71.0589  645966
## 2 2      Hartford 41.7627 -72.6743  125017
## 3 3 New York City 40.7127 -74.0059 8406000
## 4 4  Philadelphia 39.9500 -75.1667 1553000
## 5 5    Pittsburgh 40.4397 -79.9764  305841
## 6 6    Providence 41.8236 -71.4222  177994
```

```r
leaflet(cities) %>%
  addTiles() %>%
  addCircles(lng = ~Long, lat = ~Lat, weight = 1,
    radius = ~sqrt(Pop) * 30, popup = ~City)
```

Errors

Sometimes your code will generate errors. Here's an example:

```r
# Source a file that doesn't exist
source("missing.R")
```

```
## Warning in file(filename, "r", encoding = encoding): cannot open file
## 'missing.R': No such file or directory
```

```
## Error in file(filename, "r", encoding = encoding): cannot open the connection
```

Notice that the line that caused the error is highlighted, and you can see the error's traceback, just as you can in the RStudio console. If an error occurs while you're running chunks, the error will cause the notebook to stop running, and the cursor will scroll to the point where the error occurred.

R Notebooks

This file is a R Notebook. It is a great way to share your analysis with others because it contains both the html output as well as the .Rmd file with the script to generate that output.

Saving and Sharing

A notebook's source code is always in an `.Rmd` file. Whenever you save it, a sidecar `.nb.html` file is generated. This file contains a rendered copy of the notebook itself. No special viewer is required.

It also contains a copy of the notebook's source `.Rmd` file.

To look at the `.nb.html` file, click *Preview* in the RStudio editor toolbar. This is a fundamental difference between notebooks and other R Markdown documents; pressing this button doesn't actually cause any of your code to run, it just shows you the HTML file already prepared. It will automatically update whenever you save the `.Rmd` file.

If you open the `.nb.html` file in a web browser, you'll see an option to download the source. You can also open an `.nb.html` file in RStudio; when you do this, RStudio will automatically extract the `.Rmd` file and outputs inside it and open the file in the notebook editor.

Notebooks as R Markdown Documents

A notebook is also an R Markdown document. Try changing the YAML header in this document so that `html_document` is the first option, then clicking `Knit` (or just pulling down the *Preview*) menu. You could also create a PDF from the notebook, a Word document, or even a dashboard.

Publishing

The notebook output is a html file with is easily shareable and can be open in any modern browser - so no R Studio required on the receiving end. R Studio also offers a service called RPubs which let's you publish documents (and shiny apps) so you can share them with others via a link.

To do so, click the **publish** button in RStudio and enter a few things (like title and description) to publish the document (and make it publicly available). You need to sign up for RPubs beforehand to be able to do this quickly.

Other output formats

R Notebooks are also R Markdown documents and as such we can choose other output formats. Try to knit the document to html and pdf by choose the down arrow next to the preview button.

You can render the same R Markdown file into several different formats. There are two ways to change a file's output format.

First, you can click the triangle icon next to "Knit HTML" at the bottom of the pane that displays your R Markdown file. This will open a drop down menu that gives you the choice of rendering as an HTML document or a pdf document.

Second, you can change the output field in the YAML block at the top of your document. For example, this YAML block will create an HTML file:

```
---
output: html_document
---
```

This one will create a pdf file:

```
---
output: pdf_document
---
```

This one will create a MS Word file:

```
---
output: word_document
---
```

And this one will create a Markdown file:

```
---
output: md_document
---
```

Specify knitr and pandoc options

Each R Markdown output template is a collection of knitr and pandoc options. You can customize your output by overwriting the default options that come with the template.

For example, the YAML header below overwrites the default code highlight style of the pdf_document template to create a document that uses the zenburn style:

```
---
title: "Demo"
output:
  pdf_document:
    highlight: zenburn
---
```

The YAML header below overwrites the default bootstrap CSS theme of the html_document template.

```
---
title: "Demo"
output:
  html_document:
    theme: spacelab
---
```

Brand your reports with style sheets

Above, I discussed a way to change the CSS style of your HTML output: you can set the theme option of html_document to one of default, cerulean, journal, flatly, readable, spacelab, united, or cosmo. (Try it out).

But what if you want to customize your CSS in more specific ways? You can do this by writing a .css file for your report and saving it in the same directory as the .Rmd file. To have your report use the CSS, set the css option of html_document to the file name, like this

Custom CSS is an easy way to add branding to your reports. The exercise folder contains a file called `faded.css`. Try out to change the appearance of your html output.