An Rmarkdown Demo

Charles J. Geyer

September 22, 2021

1 License

2 R

The version of R used to make this document is 4.1.0. The version of the `rmarkdown` package used to make this document is 2.10. The version of the `knitr` package used to make this document is 1.34.

3 Introduction

This is a demo for using the R package `rmarkdown`. To get started make a plain text file (like this one) with suffix `.Rmd`, and then turn it into a PDF file using the R commands

```
library("rmarkdown")
render("baz.Rmd", output_format="pdf_document")
```

If instead you wish to make an HTML document, change `"pdf_document"` to `"html_document"`. If instead you wish to have some other output format, how to do that is explained in the Rmarkdown documentation.

Now include R in our document. Here's a simple example

```
2 + 2
```

```
## [1] 4
```

This is a "code chunk" processed by `rmarkdown`. When `rmarkdown` hits such a thing, it processes it, runs R to get the results, and stuffs the results (by default) in the file it is creating. The text between code chunks is markdown, a "lightweight markup language" that has become widely used in several variants (it is used by both `reddit` and `github`, for example). The web site for the R variant is http://rmarkdown.rstudio.com/.

4 Plots

4.1 Make Up Data

Plots get a little more complicated. First we make something to plot (simulate regression data).

```
n <- 50
x <- seq(1, n)
a.true <- 3
b.true <- 1.5
y.true <- a.true + b.true * x
s.true <- 17.3
```

```
y <- y.true + s.true * rnorm(n)
out1 <- lm(y ~ x)
summary(out1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.266 -12.024  -5.549  11.400  49.479
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1161     4.9356   0.226    0.822
## x             1.5301     0.1684   9.083 5.35e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.19 on 48 degrees of freedom
## Multiple R-squared:  0.6322, Adjusted R-squared:  0.6245
## F-statistic: 82.51 on 1 and 48 DF,  p-value: 5.348e-12
```

4.2 Figure with Code to Make It Shown

4.2.1 One Code Chunk

```
plot(x, y)
abline(out1)
```

4.2.2 Two Code Chunks

Sometimes we want to show the code, discuss it, and then show the figure. Or for some other reason we don't want the code immediately followed by the figure. This shows how to do that.

The following figure is produced by the following code

```
plot(x, y)
abline(out1)
```

(This code doesn't actually do anything because we used the optional argument `eval=FALSE` on this code chunk.) We could omit this showing of the code if we want.

Then a hidden code chunk makes the figure.

4.3 Figure with Code to Make It Not Shown

For this example we do a cubic regression on the same data.

```
out3 <- lm(y ~ x + I(x^2) + I(x^3))
summary(out3)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3))
##
## Residuals:
```

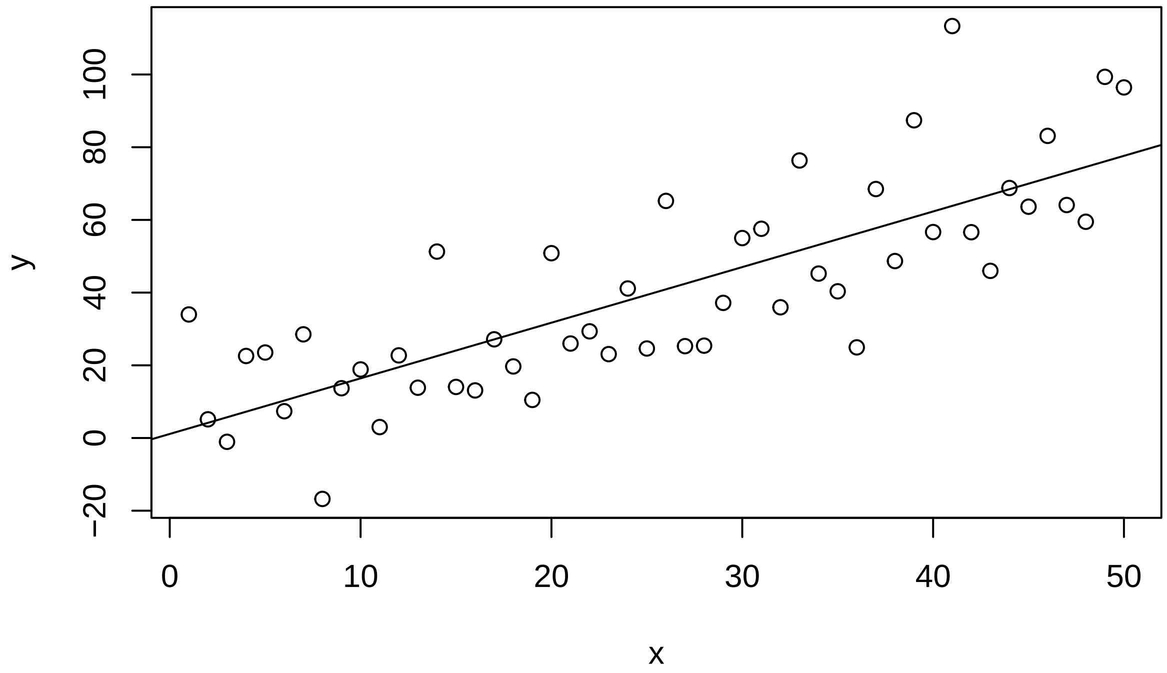

Figure 1: Scatter Plot with Regression Line

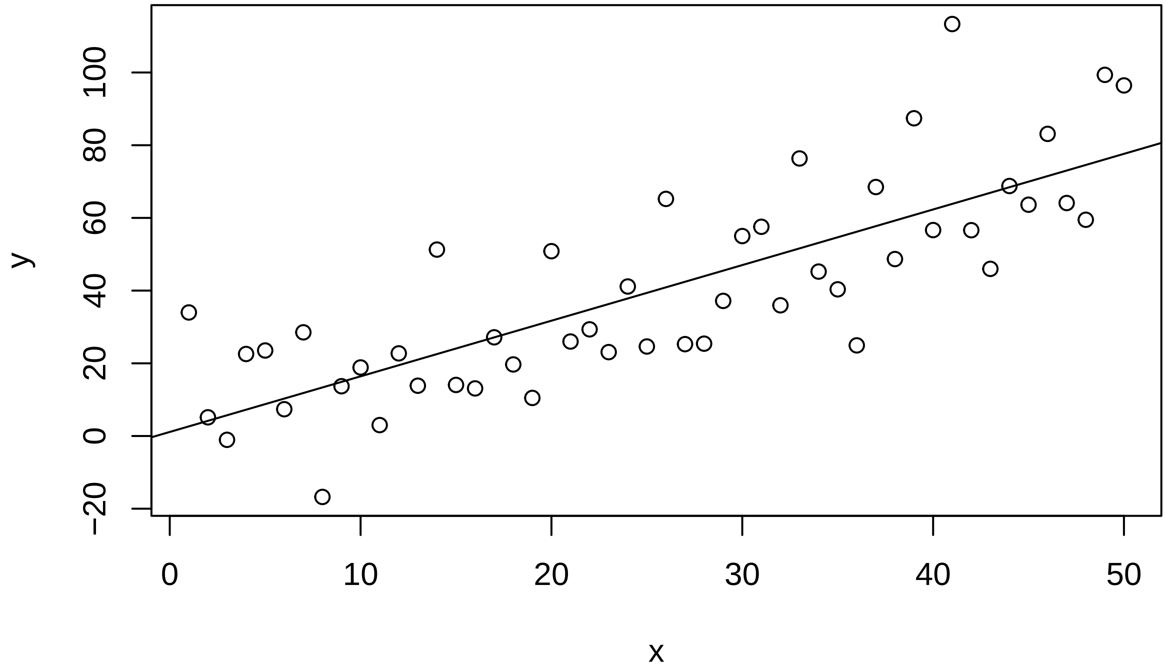

Figure 2: Scatter Plot with Regression Line

```
##     Min      1Q  Median      3Q     Max
## -31.623 -11.126  -4.307  10.509  47.557
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.089413  10.355702   1.457    0.152
## x           -0.524515   1.741136  -0.301    0.765
## I(x^2)       0.066641   0.078909   0.845    0.403
## I(x^3)      -0.000578   0.001018  -0.568    0.573
##
## Residual standard error: 16.95 on 46 degrees of freedom
## Multiple R-squared:  0.6572, Adjusted R-squared:  0.6348
## F-statistic:  29.4 on 3 and 46 DF,  p-value: 9.112e-11
```

Then we plot this figure with a hidden code chunk (so the R commands to make it do not appear in the document).

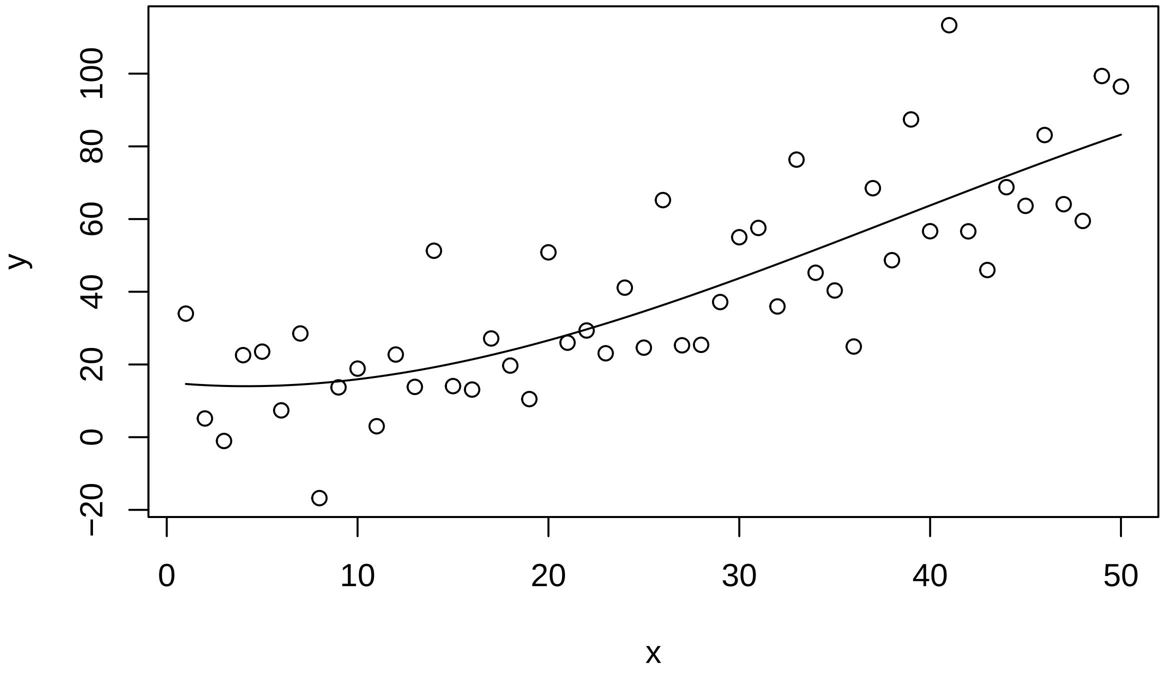

Figure 3: Scatter Plot with Cubic Regression Curve

Also note that every time we rerun `rmarkdown` these two figures change because the simulated data are random. Everything just works. This should tell you the main virtue of `rmarkdown` It's always correct. There is never a problem with stale cut-and-paste.

5 R in Text

This section illustrates how `rmarkdown` can be used to have running text computed by R.

We show some numbers calculated by R interspersed with text. The quadratic and cubic regression coefficients in the preceding regression were $\beta_2 = 0.0666$ and $\beta_3 = -0.0006$ Magic! See the source `baz.Rmd` for how the magic works.

In order for your document to be truly reproducible, you must never cut-and-paste anything R computes. Always have R recompute it every time the document is processed, either in a code chunk or with the technique illustrated in this section.

6 Tables

The same goes for tables. Here is a “table” of sorts in some R printout.

```
out2 <- lm(y ~ x + I(x^2))
anova(out1, out2, out3)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ x + I(x^2)
## Model 3: y ~ x + I(x^2) + I(x^3)
##   Res.Df   RSS Df Sum of Sq      F  Pr(>F)
## 1     48 14182
## 2     47 13310  1    871.21 3.0319 0.08833 .
## 3     46 13218  1     92.70 0.3226 0.57282
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We want to turn that into a table in output format we are creating. First we have to figure out what the output of the R function `anova` is and capture it so we can use it.

```
foo <- anova(out1, out2, out3)
class(foo)
```

```
## [1] "anova"      "data.frame"
```

So now we are ready to turn the matrix `foo` and the simplest way to do that seems to be the `kable` option on our R chunk

Table 1: ANOVA Table

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
48	14182				
47	13311	1	871	3.03	0.088
46	13218	1	93	0.32	0.573

7 Reusing Code Chunks

Code chunks can quote other code chunks. Doing this is an example of following the DRY/SPOT rule (Wikipedia articles Don’t Repeat Yourself and Single Point of Truth).

It has already been illustrated above in the section about plotting figures and showing the code in two different code chunks, but it can also be used with any code chunks

Make some data

```
x <- rnorm(100)
```

and then do something with it

```
summary(x)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## -1.876245 -0.673015  0.046993  0.008887  0.653920  2.823732 
```

and then make some other data

```
x <- rnorm(50)
```

and then do the same thing again following the DRY/SPOT rule

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -1.8752 -0.7646 -0.2919 -0.2025  0.4332  2.5295 
```

8 Summary

Rmarkdown is terrific, so important that we cannot get along without it or its older competitors `Sweave` and `knitr`.

Its virtues are

- The numbers and graphics you report are actually what they are claimed to be.
- Your analysis is reproducible. Even years later, when you've completely forgotten what you did, the whole write-up, every single number or pixel in a plot is reproducible.
- Your analysis actually works—at least in this particular instance. The code you show actually executes without error.
- Toward the end of your work, with the write-up almost done you discover an error. Months of rework to do? No! Just fix the error and rerun Rmarkdown. One single problem like this and you will have all the time invested in Rmarkdown repaid.
- This methodology provides discipline. There's nothing that will make you clean up your code like the prospect of actually revealing it to the world.

Whether we're talking about homework, a consulting report, a textbook, or a research paper. If they involve computing and statistics, this is the way to do it.