Sample Rtex document

Statistics and Statistical Programming
(MTS 525)

Your Name
youremail@u.northwestern.edu

Spring, 2019

Abstract

There are many great ways to do reproducible research. One tool is knitr; it helps you source data and execute R code directly in the (LaTeX) report you are writing. This document gives a few examples of how it is used.

1 Overview

Knitr supports the integration of R and LaTeX. You might think of it as an analogous tool to RMarkdown in this respect: whereas RMarkdown enables you to create notebooks that integrate R code and Markdown, knitr allows you to create documents that combine R code and LaTeXcode. This means that you can use knitr to create papers and reports that are fully reproducible in the sense that you can generate all the analysis and visualizations each time you create a new version of the pdf. Update your dataset? Collect new data? Discover a bug in the analysis code? If you work on your project in knitr (or RMarkdown), you can quickly regenerate the analysis, tables, and figures with a single click.

We will discuss some of the advantages and tradeoffs of this approach, as well as some more general thoughts about reproducible research workflows in class. The rest of this document illustrates how you might go about creating a knitr document that presents analysis of some data. We have

created it in Overleaf, a web-based LaTeXcollaboration and editing tool that our research group uses frequently. Keep in mind, this document is just a regular LaTeXdocument with the knitr features (and some other odds and ends) enabled within it. That means you can still do all the other things LaTeXcan do on top of the knitr features (e.g, manage styling and references).

2 Tables

You can do lots of things within a document. For example, here I create a dataframe and a table of the first few rows. Also, notice that in the caption I use the `\Sexpr{}` syntax, which lets you access R variables inline within a document.

x.1	x.2	x.3	y
-0.8593736	-0.0830516	FALSE	-1.480205
-1.1634803	0.6711977	FALSE	-1.342805
2.9916422	-0.2980747	FALSE	5.908999
2.2765654	0.1153419	FALSE	5.588834
-0.4945474	0.6378512	TRUE	1.173707
1.5567156	1.9355731	FALSE	7.685112

Table 1: This is a test table. The mean value of y is 0.5185692.

3 Code

You can also display code, by changing 'echo' to TRUE. In this example, I get the mean of y for each $x.3$ group.

```
# Get the mean for each group
aggregate(y ~ x.3, data=df, mean)

##     x.3          y
## 1 FALSE -0.4523029
## 2  TRUE  1.5290688
```

4 Regression Tables

You may also want to display regression tables. This simple table has two models: one of just $x.1$ as a predictor for y. The second model includes the other predictors.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.2441	0.3381	0.72	0.4720
x.1	1.8842	0.3192	5.90	0.0000

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0101	0.1395	0.07	0.9426
x.1	2.0417	0.0920	22.18	0.0000
x.2	2.9762	0.0967	30.78	0.0000
x.3TRUE	1.0451	0.1974	5.29	0.0000

5 Plots

You can also dispaly plots. You just put the ggplot code in a code chunk, include a fig.height or fig.width to change the dimensions. Calling it within a "figure" environment (as I do here) allows additional control.

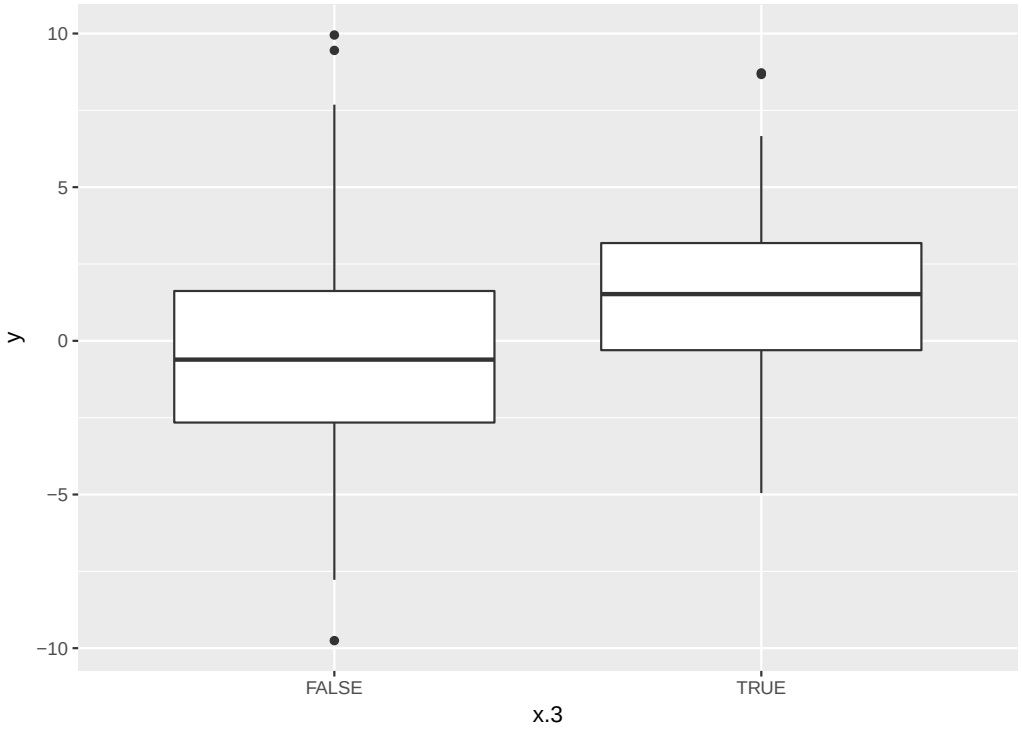

Figure 1: Boxplot of the values of y for each x.3 group

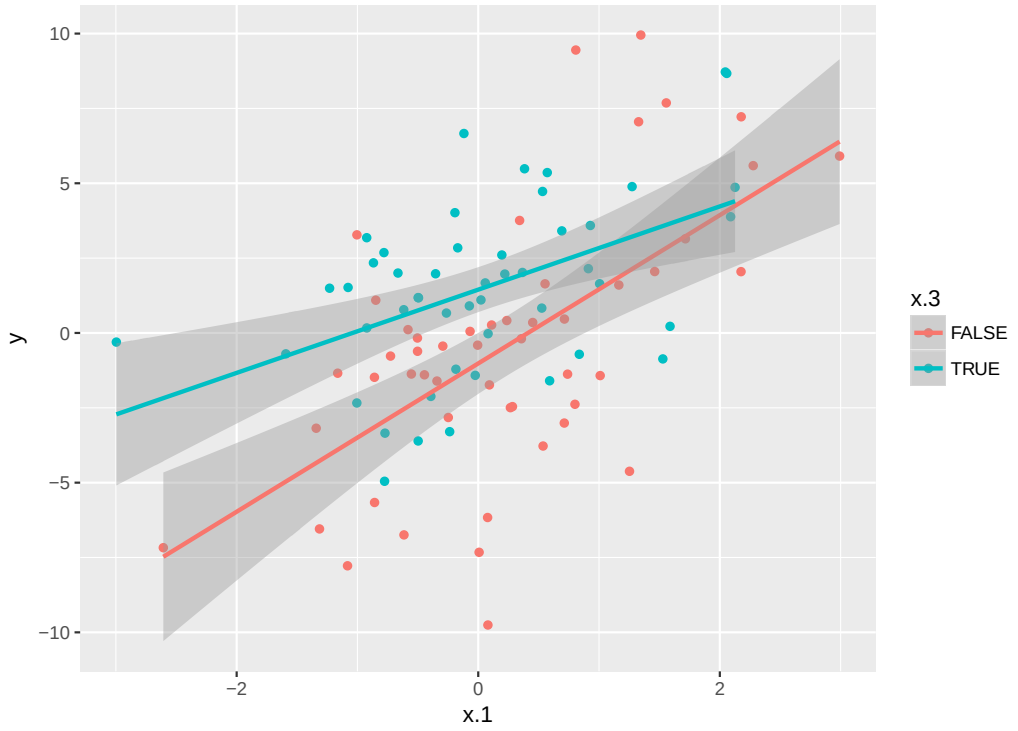

Figure 2: Scatterplot of y by x.1, grouped by x.3. Lines give the linear regression of y on x.1 for each group.