You Can Save More Time from Homework/Projects/Data Analysis for Beer/Fun/Whatever

Yihui Xie[1]

[1]Department of Statistics, Iowa State University

Abstract

Life is not easy for an applied statistician due to endless technological challenges. One statistician once told me one solution would be to "become a theoretician where the worst technological challenge one will face is sharpening one's pencil".

So you spent 80% of your time on obtaining and cleaning the data from a variety of sources, using technologies like HTML, XML, JSON, MySQL, MS Excel (seriously?), R, awk, grep, sed, Python... you name it. You clicked a lot of buttons and menus. You wrote code here and there. As the deadline is approaching, you quickly made a screenshot of the plot window in R that eventually became Figure 8 in your report.

The report is finished. There is still one hour left. Great. The sun is so bright, and air so fresh. But, but you suddenly realized they (whoever "they" are) just corrected a few numbers in the data source. Then echoing in your mind are the most miserable words in the world, ever:

Do what you did again!

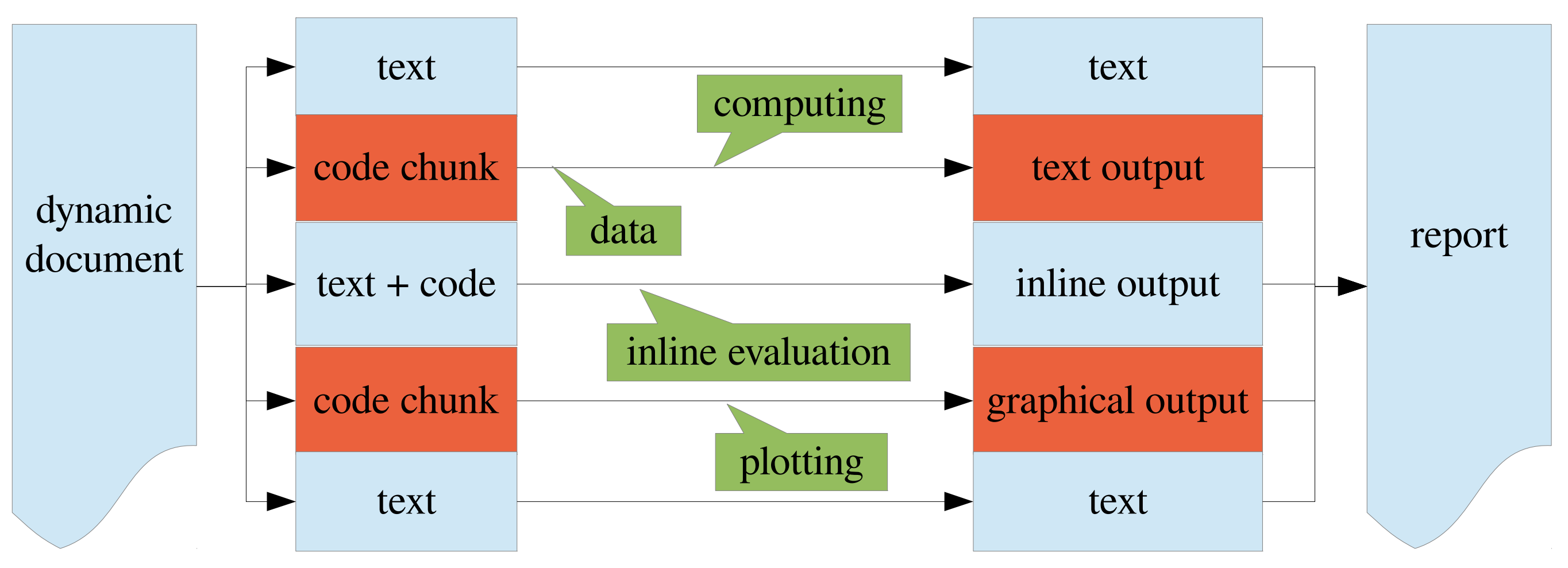
Typing, instead of Clicking

- it may be faster to click once than typing a command once, but
- it will be $+\infty$ faster to run a command $+\infty$ times than clicking buttons
- it is not easy to remember which buttons you clicked (ask yourself 6 months later)
- commands can faithfully record whatever you did, and a computer will not be bored by running a script a thousand times
  (you can always check if there are mistakes in the commands, but you cannot easily recall mistakes if you clicked buttons)
- the source code is real! (no matter how cute the output is)

Reproducible Research and Literate Programming

- if an analysis is not reproducible, it is *probably* problematic
  (if it is reproducible, it does not necessarily mean it is correct)
- it is easier to reproduce results via code than human memory
- traditional programming is almost always only about code, and literate programming = code + narratives (in one document)
  - idea conceived by Donald Knuth (1984); original implementation was TEX (narratives) + Pascal (computing), which did not seem to become popular
  - Sweave (F. Leisch, 2002) ≈ LATEX + R; it proved tremendously helpful to statisticians, but LATEX is an unnecessary obstacle
  - **knitr** (Y. Xie, 2013) = whatever + whatever (in theory), e.g., LATEX + R, HTML + shell scripts, Markdown + R, and so on
- now you embed code in reports, e.g., you write "the slope of the regression model is `` `r coef(fit)[2]` ``" instead of "the slope [...] is `3.9324`"; no matter how the model `fit` changes, you always get the correct slope value in the report
- it may take other people 2,000 hours to find out what you did because they could not reproduce your findings and you did not provide source code, e.g., the Duke Saga (`http://www.economist.com/node/21528593`; patients might have lost their lives due to false findings)

Dynamic Documents



A dynamic document is a mixture of source code chunks and narratives. The narratives describe what you want to do, and the code does what you really do. The output is generated automatically and dynamically from the dynamic document.

A Minimal (Markdown) Example

~~~
We examine the relationship between speed and stopping
distance using a linear regression model:
$Y = \beta_0 + \beta_1 x + \epsilon$.

```{r model, fig.width=5, fig.height=4}
par(mar = c(4, 4, 1, 1), mgp = c(2, 1, 0), cex = 0.8)
smoothScatter(cars, pch = 20, col = 'gray10')
fit <- lm(dist ~ speed, data = cars)
abline(fit, lwd = 2)
```

The slope of the linear regression is `r coef(fit)[2]`.
~~~

How to Compile It

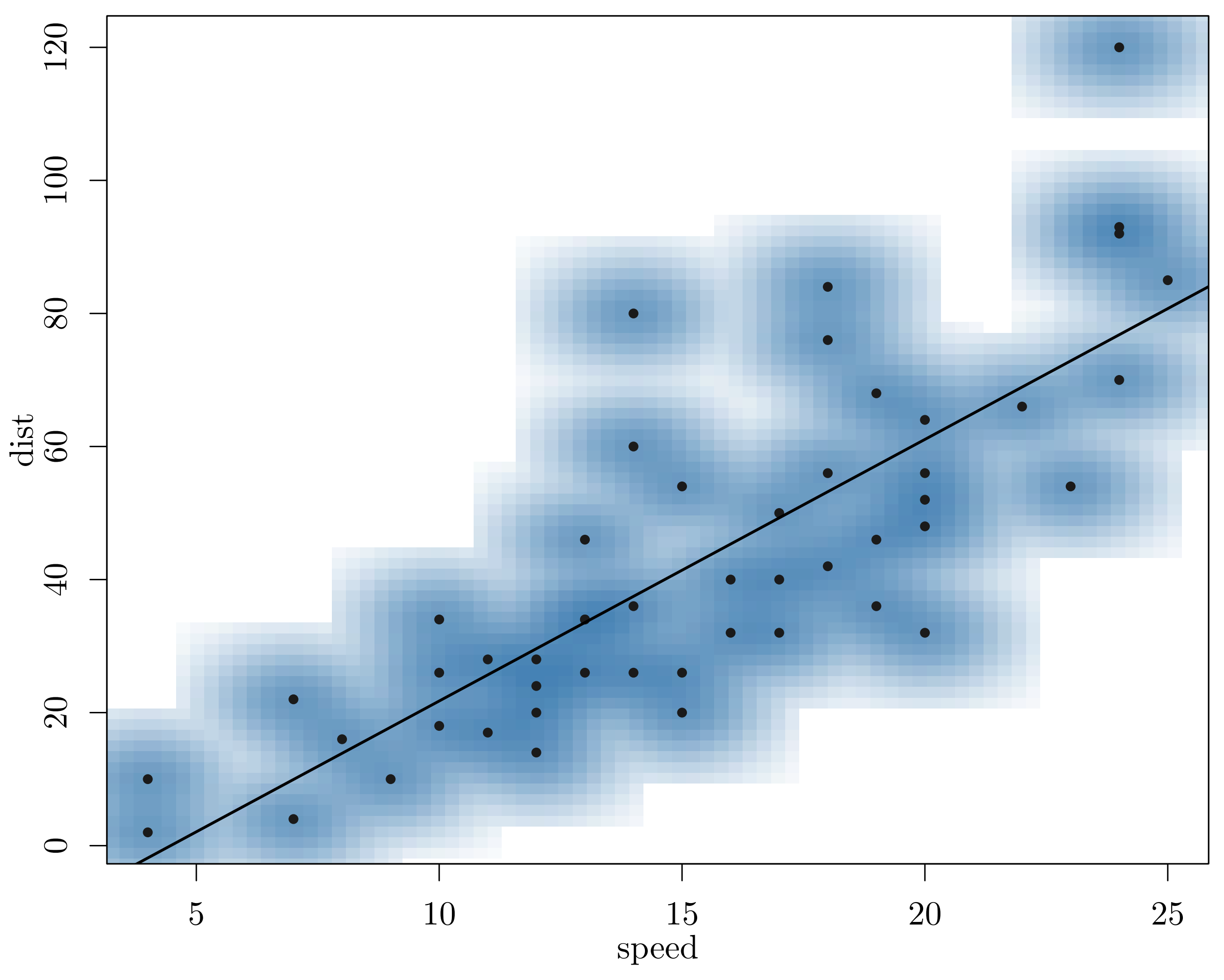
The easiest way is to save the above file in RStudio (`http://www.rstudio.com`) as, say, example.Rmd, and click the button `Knit HTML`, which essentially calls the **knitr** package in R to compile the document, so you have to install this package first: `install.packages('knitr')`

You can use any other editors such as Emacs/ESS, LyX, Vim, TEXShop, TEXmaker, and so on. The key is to call the function `knit()` in **knitr**.

Output of the Minimal Example

We examine the relationship between speed and stopping distance using a linear regression model: $Y = \beta_0 + \beta_1 x + \epsilon$.

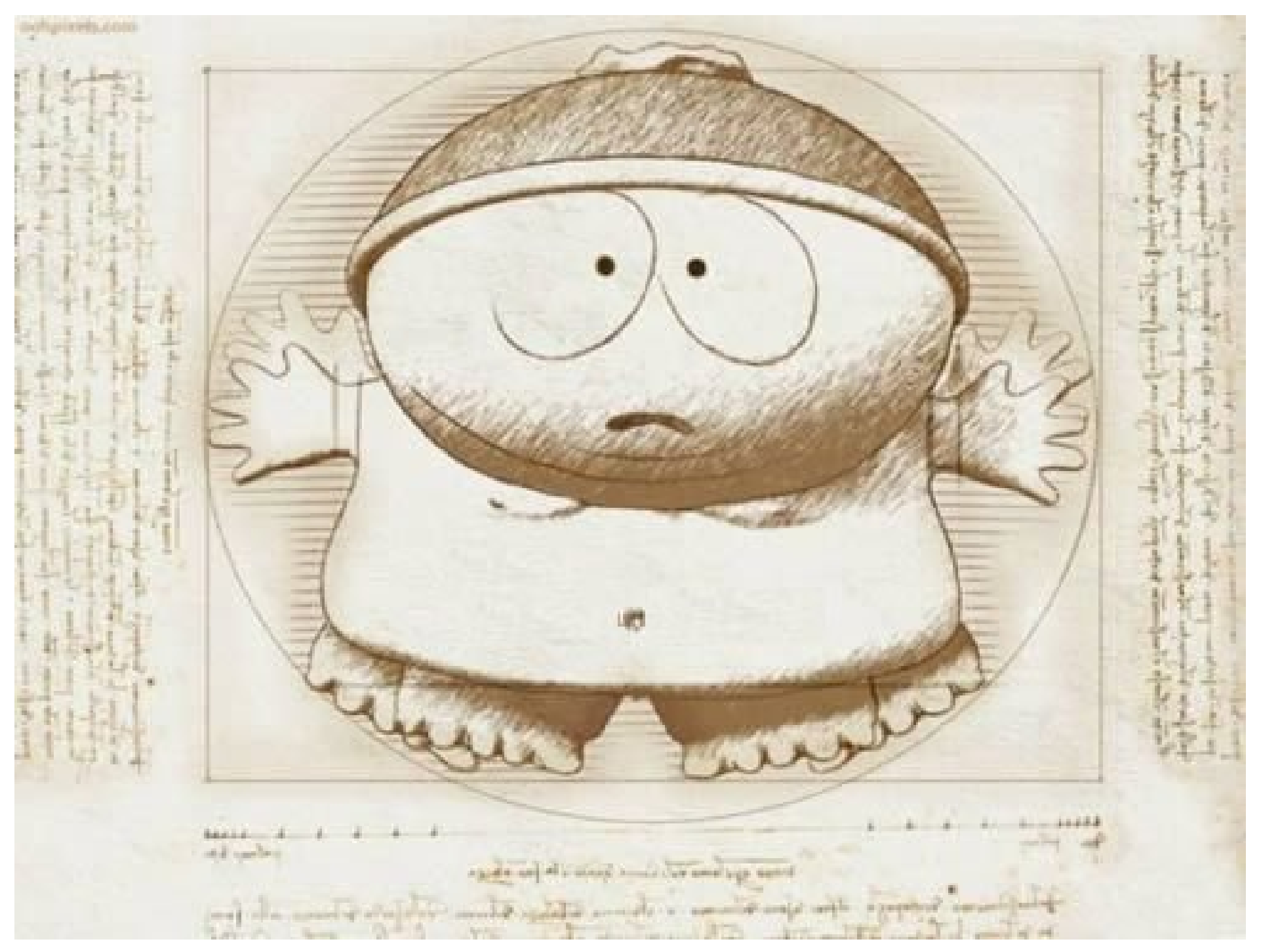
```
par(mar = c(4, 4, 1, 1), mgp = c(2, 1, 0), cex = 0.8)
smoothScatter(cars, pch = 20, col = "gray10")
fit <- lm(dist ~ speed, data = cars)
abline(fit, lwd = 2)
```



The slope of the linear regression is 3.9324.

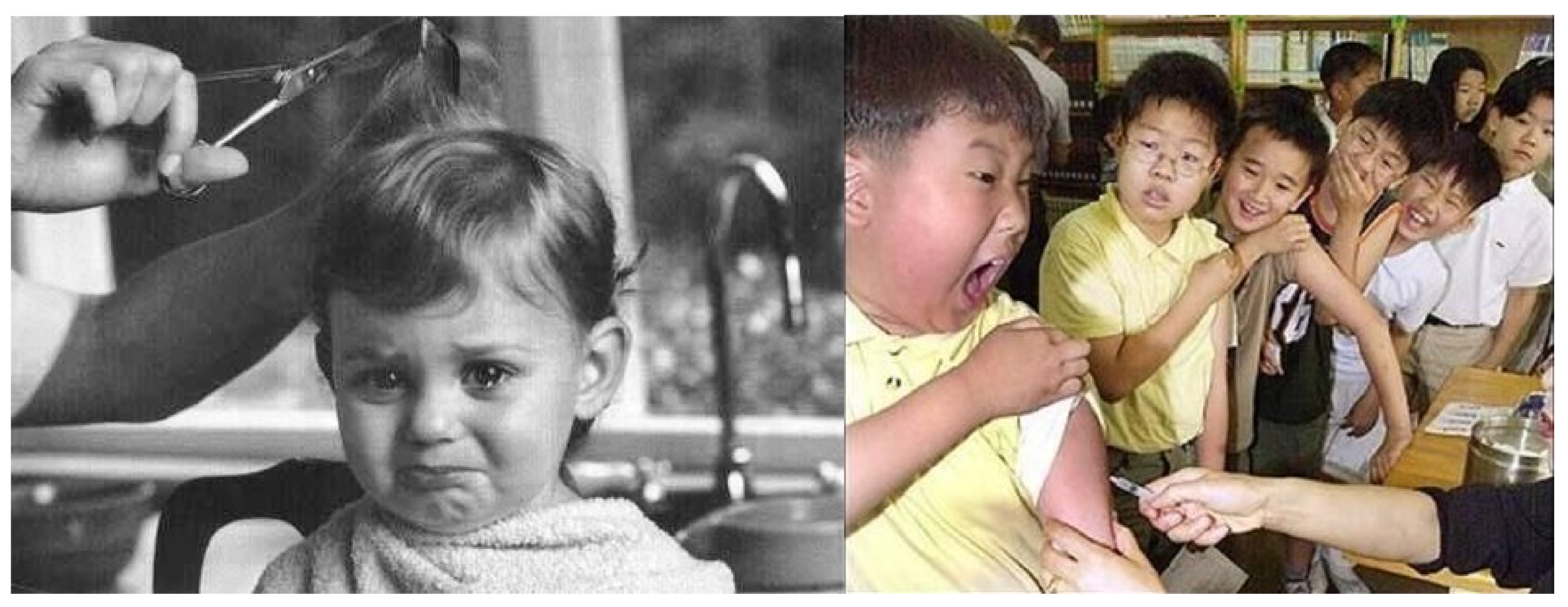
Selected Features

- full support of (LATEX | HTML | Markdown | reST | AsciiDoc) + R
- fine control of output, e.g., show/hide source code, adjust figure sizes, ...
- cache for time-consuming computations
- child documents for large reports
- limited support of foreign languages like C++, Python, Ruby, shell scripts, and Julia, etc



There are a large number of options that you can tune until the output fits your report, and you almost never need to copy and paste any results between different programs (e.g., from R to LATEX).

Change Your Workflow



It hurts when I take Excel away from you, but it is worth it in the long run (like more beautiful hair and a healthier body).

For More Information

The **knitr** website has the comprehensive documentation: `http://yihui.name/knitr`; the **knitr** book (`http://amzn.com/1482203537`) is just a summary of the website and many other questions asked by users in the past two years. Please join the development of this package on GitHub: `https://github.com/yihui/knitr`

Summary

- it is not only about reproducible research, but also easier/less work (laziness is the ultimate motivation of the universe)
- it only takes a few minutes to get started with the more reliable way to do data analysis (see `http://rpubs.com` for thousands of shared examples; nobody requested them to do so)
- if you are required to "do it again", it takes almost no additional effort (simply run `knit()` again or click one button in the editor)

Contact

Yihui Xie
Email: `xie@yihui.name`
Homepage: `http://yihui.name`
Twitter: `@xieyihui`; GitHub: `@yihui`

This work was not supported by any beer companies. They will regret it some day, of course.