improveR

Contents

1 Introduction

1.1 Blurb

This short course covers the core skills required for a budding R user to develop a strong foundation for data analysis in the RStudio environment. Within the framework of a reproducible research workflow we will cover importing and cleaning data, efficient coding practices, writing your own functions and using the powerful `dplyr` data manipulation tools.

1.2 Key Topics

- Reproducible Research
- R Studio and project management
- Importing and cleaning data
- Good coding practices in R
- standard control structures
- Vectorisation and `apply` functions
- Writing your own funcitons
- Data manipulation with `dplyr`
- Piping/chaining commands

1.3 Course information

Intended audience Anyone interested in quantitative data analysis using open source tools.

Prior knowledge Knowledge of R (as covered in R: An introduction).

Resources Course handbook

Software RStudio & R 3.1.2

Format Presentation with practical exercises

Where next? R:

2 Reproducible Research [presentation only]

Reproducible reseach means making the data and the code of our analysis available in a way that is sufficient and easy for an independent researcher to recreate our findings.

This is the golden standard of scientific inquiry, and is increasinlgy and rightly becoming a requirement in academic publishing, and by funding bodies.

It is also a way of establishing better working habits, reduce the potential for error, develop a more streamlined research process, and make for easier collaboration.

Reproducible reseach does take a bit of upfront investment in learning the tools and setting up your workflow. Luckily RStudio has integrated many of the tools required in one platform, making it easier than ever to

2.1 Why?

- Reinhart Rogoff Excel spreadsheet

Document everything! This means never running any code from the command prompt, always writing it into a script file and running it from there.

3 Set-up [presentation and practical]

3.1 RStudio

3.2 Project management

A crucial requirement for conducting reproducible research, and one that has to be carefully considered before you embark on your analysis, is your plan on how the data, code and outputs will be organised. The project management structure proposed here is just a suggestion, and you should adapt it to your specific needs, but it is highly recommended that you stick to one such system consistently, instead of comming up with 'ad hoc' solutions for every new project.

RStudio makes it extremely easy to divide your work into separate projects, allowing you to neatly organize and acces your work.

3.3 Literate programming

3.3.1 Consistent coding style e.g.:

- Google's R Style Guide
- Hadley Wickham's Style Guide

3.3.2 Commenting

3.4 * Bonus section: github

3.5 PRACTICAL: new R project

- personalise RStudio settings (don't save .Rdata etc)
- new project folder with subfolders (data, figures, scripts)
- new Rproject

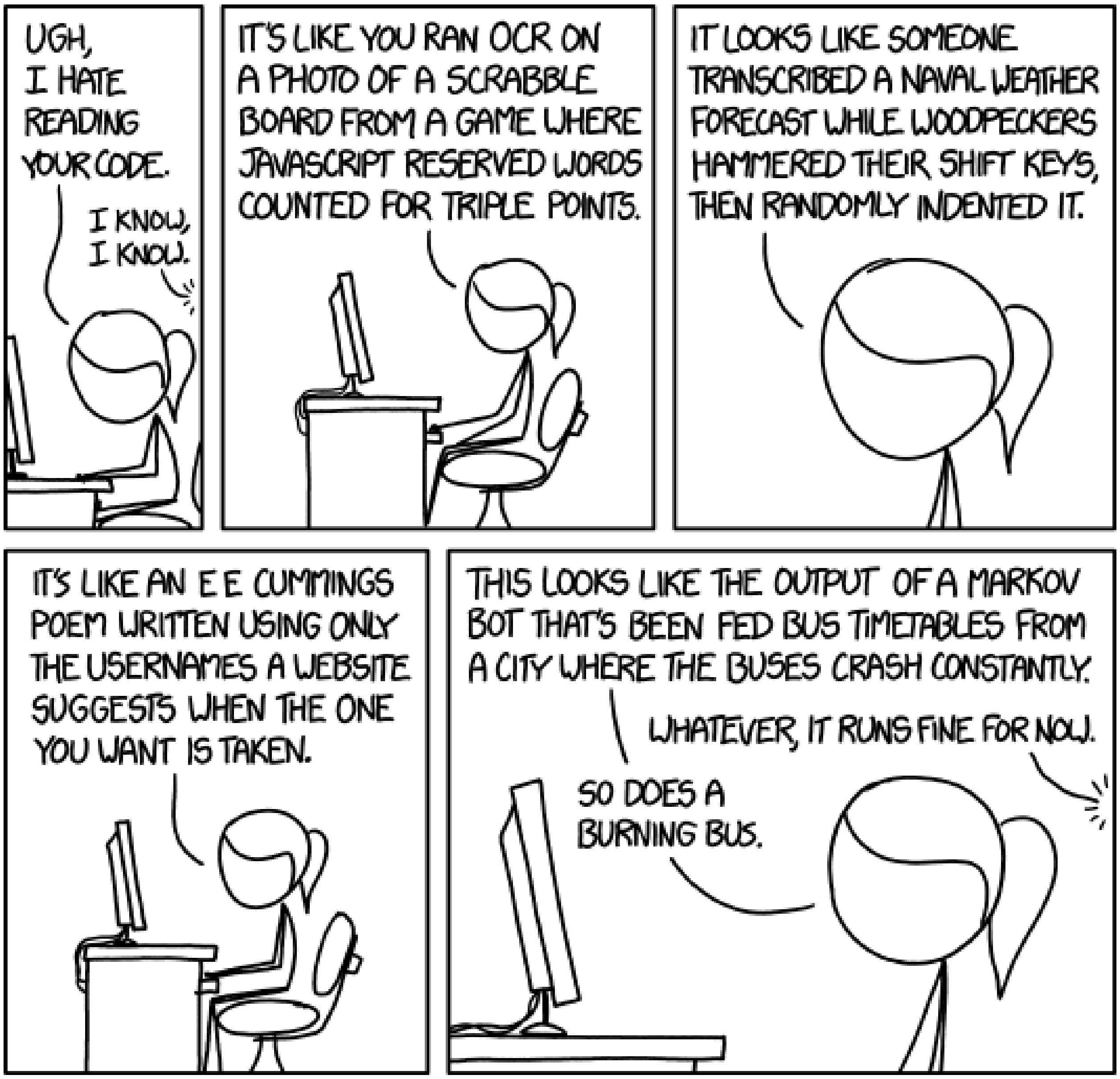
UGH, I HATE READING YOUR CODE.
I KNOW, I KNOW.
IT'S LIKE YOU RAN OCR ON A PHOTO OF A SCRABBLE BOARD FROM A GAME WHERE JAVASCRIPT RESERVED WORDS COUNTED FOR TRIPLE POINTS.
IT LOOKS LIKE SOMEONE TRANSCRIBED A NAVAL WEATHER FORECAST WHILE WOODPECKERS HAMMERED THEIR SHIFT KEYS, THEN RANDOMLY INDENTED IT.
IT'S LIKE AN E E CUMMINGS POEM WRITTEN USING ONLY THE USERNAMES A WEBSITE SUGGESTS WHEN THE ONE YOU WANT IS TAKEN.
THIS LOOKS LIKE THE OUTPUT OF A MARKOV BOT THAT'S BEEN FED BUS TIMETABLES FROM A CITY WHERE THE BUSES CRASH CONSTANTLY.
WHATEVER, IT RUNS FINE FOR NOW.
SO DOES A BURNING BUS.

Figure 1: title

4 Workflow

4.1 Importing data

The original data should be read-only!!

- url
- unzip
- (colClasses)
- APIs ## Data tidying
- gather/ spread

4.2 PRACTICAL: Import and clean some data

- download and import data
- do some `tidyr` stuff with it
- think about commenting and file structure!

5 Efficient Coding

5.1 Standard control structures

5.1.1 Conditional execution

5.1.2 Looping

5.1.3 PRACTICAL

5.2 Vecotrisation and `apply` family of funcitons

5.2.1 PRACTICAL

benchmarking apply vs for loops

5.3 Writing your own functions

5.3.1 objects, types, environments

5.3.2 passing arguments

5.3.3 PRACTICAL

5.4 Data manipulation with `dplyr`

5.4.1 Subsetting

- `filter`
- `sample`
- `slice`
- `distinct`
- `select`

5.4.2 Grouping

- `group_by`

5.4.3 Summarizing

- with own function

5.4.4 Making new variables

- `mutate`

5.4.5 Piping/chaining daisies

5.4.6 PRACTICAL

5.5 FINAL PRACTICAL

something along the lines of:

- Fun1: a function to be called in summarize or mutate (e.g. z-score)
- Fun2: a chain (that calls Fun1), and then filters the table in some way e.g. subset for each country
- Fun3: a nice plotting function that takes the result of Fun2 and plots it, using `paste()` for titles etc..

5.6 Accessing Data Using APIs

APIs (Application Programming Interface) allow standardised data access to a variety of web resources. More and more websites are publishing them making it easy for developers and researchers to dynamically access or update content.

> When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

[Source: WIkipedia]

With R we can easily handle both processes:

- Input: The HTTP request
- Output: The .json or .xml response

We will use the `httr` package to