Coding data science

March 2019

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio
What is data analysis?
Data analysis is the process by which data becomes understanding, knowledge, and insight.
Data analysis is the process by which data becomes understanding, knowledge and insight.
Import

Consistent way of storing data
Import

Tidy

Consistent way of storing data

Understand
Import

Tidy
Consistent way of storing data

Transform
Create new variables & new summaries

Model
Scales, but doesn’t (fundamentally) surprise

Visualise
Surprises, but doesn’t scale
Tidy: Consistent way of storing data

Import

Transform: Create new variables & new summaries

Visualise: Surprises, but doesn’t scale

Model: Scales, but doesn’t (fundamentally) surprise

Communicate

Automate
What is data science?
Data science = data analysis + programming
Tidy

Consistent way of storing data

Import

Surprises, but doesn't scale

Transform

Create new variables & new summaries

Visualise

Model

Scales, but doesn’t (fundamentally) surprise

Automate

Communicate

Program
Import

Tidy

Transform

Visualise

Model

Communicate

Automate
Why code?
The disadvantages of code are obvious
Why code?

1. Code is text
2. Code is readable
3. Code is reproducible
Top Questions

0 votes 0 answers 2 views
Who to scroll to in a react app when route changes for screen reader
react react-router accessibility react-router-dom asked 55 secs ago dagst1 9,694

0 votes 0 answers 2 views
What if I schedule tasks for celery to perform every minute and it is not able to complete it in time?
celery scheduled-tasks celery-task celerybeat asked 1 min ago rav 1

1 vote 0 answers 5 views
Gerrit: is there a way to push directly into master?
gerrit modified 2 mins ago leoyulwah 1,909

0 votes 0 answers 3 views
kubectl run-ing a tarball'd image
docker kubernetes kubectl asked 2 mins ago adolberto 4,470

0 votes 0 answers 7 views
Asp.Net Identity RequireUniqueEmail = false throws exception on CreateAsync
asp-net-identity unique own modified 2 mins ago Dmitry Duka 1

0 votes 0 answers 3 views
Square: Call to undefined function charge
square-connect asked 3 mins ago John 4,104

0 votes 0 answers 4 views
Python - Social Studio API - How to unpack JSON into pandas data frame
python json pandas asked 3 mins ago Ulises Sotomayor 36

2 votes 1 answer 15 views
Issues running airflow scheduler as a daemon process
python amazon-ec2 ubuntu-18.04 airflow apache-airflow answered 3 mins ago Tagar 2,393

1 vote 1 answer 20 views
Flutter animation how to fade in/out gradually
answered 3 mins ago Coffee_Indones 8,887
1. Code is text
2. Code is readable
3. Code is reproducible
Why code?

1. Code is text
2. Code is read-able
3. Code is reproducible
.Rmd

Prose and code

.md

Prose and results

.html

Human shareable
What about non-programmers?
You don’t need to be a programmer to code!
Your turn: What data do we need to recreate this plot?
## Underlying data

```r
# A tibble: 193 x 6

<table>
<thead>
<tr>
<th>country</th>
<th>four_regions</th>
<th>year</th>
<th>income</th>
<th>life_exp</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>asia</td>
<td>2015</td>
<td>1750</td>
<td>57.9</td>
<td>33700000</td>
</tr>
<tr>
<td>Albania</td>
<td>europe</td>
<td>2015</td>
<td>11000</td>
<td>77.6</td>
<td>2920000</td>
</tr>
<tr>
<td>Algeria</td>
<td>africa</td>
<td>2015</td>
<td>13700</td>
<td>77.3</td>
<td>39900000</td>
</tr>
<tr>
<td>Andorra</td>
<td>europe</td>
<td>2015</td>
<td>46600</td>
<td>82.5</td>
<td>78000</td>
</tr>
<tr>
<td>Angola</td>
<td>africa</td>
<td>2015</td>
<td>6230</td>
<td>64</td>
<td>27900000</td>
</tr>
<tr>
<td>Antigua and Barbuda</td>
<td>americas</td>
<td>2015</td>
<td>20100</td>
<td>77.2</td>
<td>99900</td>
</tr>
<tr>
<td>Argentina</td>
<td>americas</td>
<td>2015</td>
<td>19100</td>
<td>76.5</td>
<td>43400000</td>
</tr>
<tr>
<td>Armenia</td>
<td>europe</td>
<td>2015</td>
<td>8180</td>
<td>75.4</td>
<td>2920000</td>
</tr>
<tr>
<td>Australia</td>
<td>asia</td>
<td>2015</td>
<td>43800</td>
<td>82.6</td>
<td>23800000</td>
</tr>
<tr>
<td>Austria</td>
<td>europe</td>
<td>2015</td>
<td>44100</td>
<td>81.4</td>
<td>8680000</td>
</tr>
</tbody>
</table>

# … with 183 more rows
```
Phonics are important!

`gapminder %>% filter(year == 2015) -> gapminder15`

- Take the gapminder data, then
- Filter rows where year equals 2015, creating the `gapminder15` variable.
gapminder15 %>%
ggplot(aes(income, life_exp))
gapminder15 %>%
ggplot(aes(income, life_exp)) + geom_point()
gapminder15 %>%
  ggplot(aes(income, life_exp)) +
  geom_point() +
  scale_x_log10()
gapminder15 %>%
  ggplot(aes(income, life_exp)) +
  geom_point(aes(colour = four_regions)) +
  scale_x_log10()
```
gapminder15 %>%
ggplot(aes(income, life_exp)) +
geom_point(aes(colour = four_regions, size = pop)) +
scale_x_log10()
```
But
df %>%
  rename(
    date = `Date Created`,
    name = Name,
    plays = `Total Plays`,
    loads = `Total Loads`,
    apv = `Average Percent Viewed`
  )

And this is painful!
RStudio Addins

**IMPORTANT NOTE:** Support for addins is available only within the most recent release of RStudio (v0.99.878 or later). If you want to try out addins please be sure to download this release.

RStudio addins provide a mechanism for executing R functions interactively from within the RStudio IDE—either through keyboard shortcuts, or through the Addins menu.

An addin can be as simple as a function that inserts a commonly used snippet of text, and as complex as a Shiny application that accepts input from the user, and later mutates a document open in RStudio. The sky is the limit!

Here are two examples of addins in action (click on the thumbnail to see a brief demonstration):

**Using Addins**

This guide will walk you through the basics of installing addins, binding keyboard shortcuts to them, and finally developing your own addins.

**Installation**

RStudio Addins are distributed as R packages. Once you’ve installed an R package that contains addins, they’ll become immediately available within RStudio.

Let’s start by playing around with a couple of the example addins provided by the *addinexamples* package. Within RStudio, install this package (plus its requisite dependencies) with:

```r
devtools::install_github("rstudio/addinexamples", type = "source")
```

**Running Addins**
What next?

df %>%
  filter(n > 1e6) %>%
  mutate(x = f(y))) %>%
  ???

# How predictable is next step from
# previous steps?
Can we do more with autocomplete?

Where do dialogs and autocomplete intersect?
Learning from examples

(a) **Reported crime in Alabama**

- **before:** {‘in’, ‘ ’} ‘Alabama’ $\rightarrow$ {‘Alabama’, word}
- **selection:** {‘Alabama’} ‘in’ $\rightarrow$ {‘in’, word, lowercase}
- **after:** $\emptyset$ ‘ ’ $\rightarrow$ {‘ ’}

(b) **before:** {(‘ ’), {‘in’, ‘ ’}, (word, ‘ ’), (lowercase, ‘ ’)}
- **selection:** {‘Alabama’}, (word)
- **after:** $\emptyset$

(d) {‘(lowercase, ‘ ’), {‘Alabama’}, ()} $\rightarrow$ /([a-z]+ (Alabama)/

**Figure 10.** Regular Expression Inference. (a) The user selects text in a cell. (b) We tokenize selected and surrounding text. For clarity, the figure only includes two neighboring tokens. For each token, we generate a set of matching labels. (c) We enumerate all label sequences matching the text. (d) We then enumerate all candidate before, selection and after combinations. Patterns that do not uniquely match the selected text are filtered (indicated by strike-through). (e) Finally, we construct regular expressions for each candidate pattern.
\[ p(x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \]

\[ p(\theta|y) \propto p(y|\theta)p(\theta) \]

\[ \int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi} \]

https://twitter.com/carroll_jono/status/914254139873361920
Fin
We wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important.

— John Chambers, “Stages in the Evolution of S”