

R4DS

Cohort 4

Wed 6:00 – 7:00 US Central

Twitter: @Rspjut

5-MINUTE ICE BREAKER

How late did you stay up on New Year's Eve?

AGENDA

- 5-Minute Ice breaker
- Quick Housekeeping Reminders
- Object Oriented Programming
- Chapter 4 & 5 Close Out
- Chapter 6 & Keyboard Shortcuts
- Chapter 1 – 6 Recap
- Q&A
- Getting Help
- Next Week

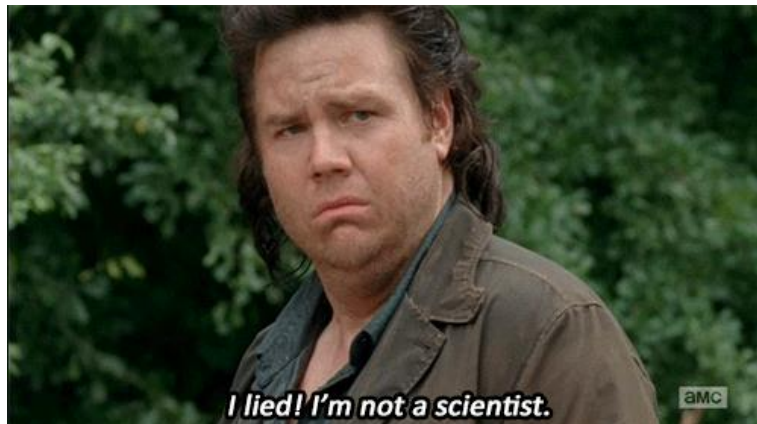
QUICK HOUSEKEEPING REMINDERS

- Video camera is optional, but encouraged.
- I purposely err on the side of going fast. Slowing me down does not hurt my feelings.
- Take time to learn the theory (Grammar of Graphics, Tidy Data whitepaper, Relational Database theory, etc.).
- Please do the chapter exercises. Second-best learning opportunity!
- Please plan on teaching one of the lessons. Best learning opportunity!

OBJECT ORIENTED PROGRAMMING

Two resources to explain object oriented programming.

- Anthony Shook (via @Scott Nestler in Slack)
- Paper by John M. Chambers

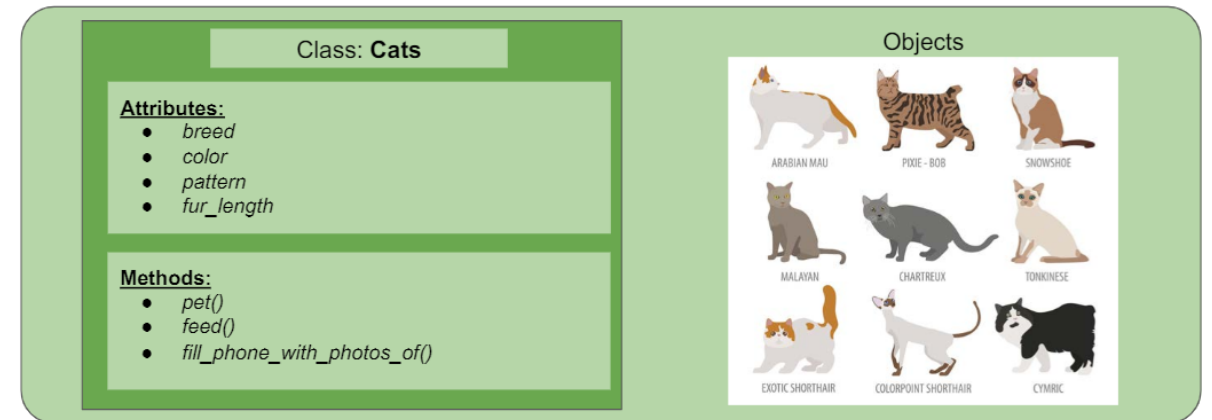


ANTHONY SHOOK

- OOP is a programming paradigm focused on **objects and their interactions**
- This is contrast to **Procedural Programming**, where you have data that you pass from function to function in order to achieve a result
- The goal in OOP isn't to provide ordered instructions but to define classes of objects with attributes and behaviors
 - And, how those attributes and behaviors change over time, or influence *other* objects!
 - Philosophically, this means classes and how they act are not separable

OOP: Classes and Methods

- A **class** is a blueprint that defines attributes and behaviors of objects
 - 'objects' are instances of 'classes' -- they are unique, but defined by the class blueprint
 - Methods and attributes are shared by objects of the class -- they apply universally*



JOHN M. CHAMBERS

Object-Oriented Programming, Functional Programming and R

John M. Chambers

Abstract. This paper reviews some programming techniques in R that have proved useful, particularly for substantial projects. These include several versions of object-oriented programming, used in a large number of R packages. The review tries to clarify the origins and ideas behind the various versions, each of which is valuable in the appropriate context.

R has also been strongly influenced by the ideas of functional programming and, in particular, by the desire to combine functional with object oriented programming.

To clarify how this particular mix of ideas has turned out in the current R language and supporting software, the paper will first review the basic ideas behind object-oriented and functional programming, and then examine the evolution of R with these ideas providing context.

Functional programming supports well-defined, defensible software giving reproducible results. Object-oriented programming is the mechanism *par excellence* for managing complexity while keeping things simple for the user. The two paradigms have been valuable in supporting major software for fitting models to data and numerous other statistical applications.

The paradigms have been adopted, and adapted, distinctively in R. Functional programming motivates much of R but R does not enforce the paradigm. Object-oriented programming from a functional perspective differs from that used in non-functional languages, a distinction that needs to be emphasized to avoid confusion.

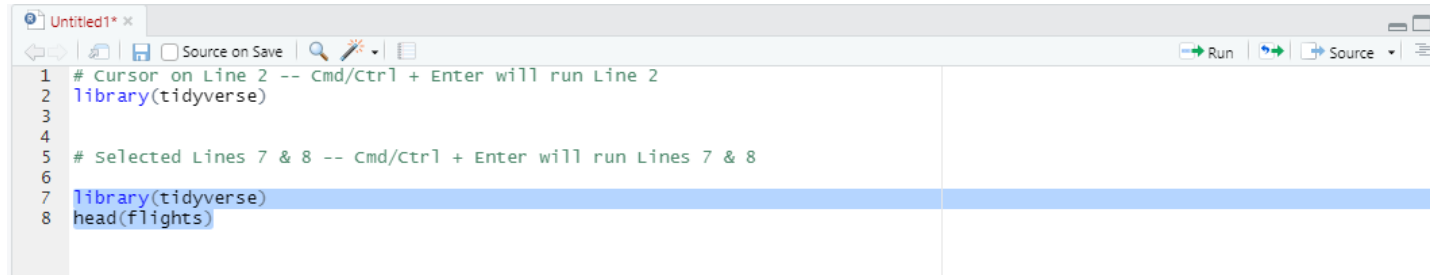
R initially replicated the S language from Bell Labs, which in turn was strongly influenced by earlier program libraries. At each stage, new ideas have been added, but the previous software continues to show its influence in the design as well. Outlining the evolution will further clarify why we currently have this somewhat unusual combination of ideas.

Key words and phrases: Programming languages, functional programming, object-oriented programming.

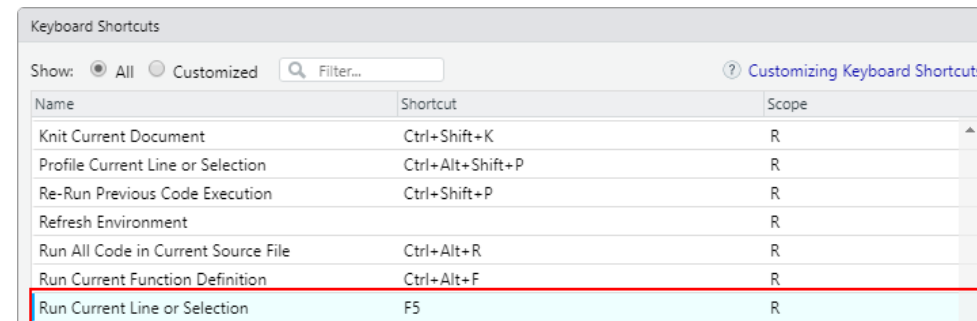
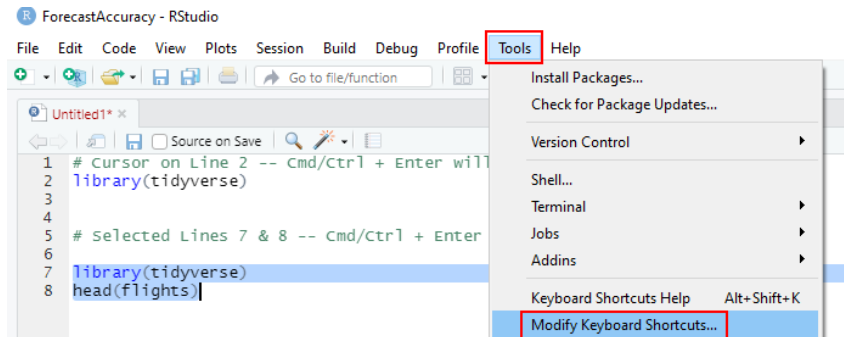
CHAPTER 4 & 5 CLOSE OUT

... Collin

CHAPTER 6 & KEYBOARD SHORTCUTS



```
1 # Cursor on Line 2 -- Cmd/Ctrl + Enter will run Line 2
2 library(tidyverse)
3
4
5 # Selected Lines 7 & 8 -- Cmd/Ctrl + Enter will run Lines 7 & 8
6
7 library(tidyverse)
8 head(flights)
```



The screenshot shows the 'Keyboard Shortcuts' dialog box. The 'Run Current Line or Selection' entry is highlighted with a red rectangle. The dialog shows a list of shortcuts with columns for Name, Shortcut, and Scope.

Name	Shortcut	Scope
Knit Current Document	Ctrl+Shift+K	R
Profile Current Line or Selection	Ctrl+Alt+Shift+P	R
Re-Run Previous Code Execution	Ctrl+Shift+P	R
Refresh Environment		R
Run All Code in Current Source File	Ctrl+Alt+R	R
Run Current Function Definition	Ctrl+Alt+F	R
Run Current Line or Selection	F5	R

Name	Symbol	R-Shortcut	Potential Shortcut	Shortcut Description
Assignment Operator	<-	Alt + -	Alt + -	Alt + Dash
Pipe Operator	%>%	Ctrl + Shift + M	Alt + .	Alt + Period

CHAPTER 1 — 6 RECAP

Top

Create

Name

- C
- C
- R is
- Assi

tidyve

Google

```
# Avoid  
x = 3  
  
# obje  
# "Obj  
x ← 3
```

- Wha

```
x  
#> [1]
```

- See

The %>% operator

The %>% (pipe) allows us to chain commands together.

As suggested by this reading, a good way to pronounce %>% when reading code is "then". ~Hadley Wickham & Garrett Grolemund (authors).

```
_smooth()  
r bands with se = FALSE  
ent colors using color = drv  
mapped to a variable so it's in the aesthetic  
tic is global to capture point and line (geom_smooth)
```

```
pg, mapping = aes(x = displ, y = hwy, color = drv)) +  
  
se = FALSE)
```



Q & A

GETTING HELP

- Ask questions during our call
- Google
- Stack Overflow
- Slack
- Office Hours r4ds.io/calendar ***
- Twitter [#rstats](https://twitter.com/rstats)
- r4ds answer keys: Jeff Arnold (preferred) or Bryan Shalloway (also good)
- Cheatsheets

NEXT WEEK...

- Chapter 7: Exploratory Data Analysis
- Look over the diamonds and mpg data

```
library(tidyverse)
```

```
?diamonds
```

```
?mpg
```

